

TUTORIAL MEMBUAT APLIKASI SISTEM MONITORING TERHADAP JOB DESK OPERATIONAL HUMAN CAPITAL (OHC)

Buku ini dibuat untuk memenuhi persyaratan kelulusan
matakuliah Program Internship I



Dibuat Oleh,
1.16.4.052 Riki Karnovi

**PROGRAM DIPLOMA IV TEKNIK INFORMATIKA
POLITEKNIK POS INDONESIA
BANDUNG
2020**

DAFTAR ISI

DAFTAR ISI.....	i
DAFTAR GAMBAR	iv
DAFTAR TABEL.....	viii
PENGANTAR	x
BAB I.....	1
DEFINISI	1
1.1 Sistem	1
1.2 Bahasa Pemrograman	28
1.3 Metode Yang Digunakan.....	29
1.4 Sistem <i>Monitoring</i>	35
1.5 <i>Job Description</i>	36
BAB II.....	54
METODELOGI PENELITIAN	54
2.1 <i>DIAGRAM ALUR PENELITIAN</i>	55
BAB III	58
PENJELASAN TOOLS DAN BAHASA PEMROGRAMAN YANG DIGUNAKAN	58
3.1 Tools Yang Digunakan.....	58
3.1.1 <i>Sublime</i>	Error! Bookmark not defined.
3.1.2 <i>XAMPP</i>	61

3.1.3 <i>Email</i>	68
3.1.4 Protokol <i>E-mail</i>	80
3.2 Bahasa Pemograman	84
3.2.1 <i>CSS</i>	84
3.2.2 <i>HTTP (HyperText Transfer Protocol)</i> dan <i>HTTPS (Hyper Text Transfer Protocol Secure)</i>	87
3.2.3 <i>HTTPS (Hyper Text Transfer Protocol Secure)</i>	88
3.2.4 <i>HTML (Hyper Text Markup Language)</i>	90
3.2.5 Hubungan <i>HTML</i> dan <i>PHP</i>	99
BAB IV	101
INSTALASI <i>TOOLS</i> YANG DIGUNAKAN	101
4.1 Instalasi <i>Tools</i> Yang Digunakan.....	101
BAB V	123
PERANCANGAN PEMBUATAN APLIKASI.....	123
5.1 Analisis	123
5.2 Perancangan.....	131
5.2.3 Kamus Alur Data.....	138
5.2.4 Struktur <i>Menu</i>	139
5.3 Perancangan Basis Data	140
5.3.1 <i>Conceptual Data Model (CDM)</i>	140
5.3.2 <i>Physical Data Model (PDM)</i>	141
5.4 Mempersiapkan <i>Tools</i>	142
5.4.1 Membuat <i>Database</i>	142

5.5 Perancangan Antarmuka	147
5.5.1 Perancangan Antarmuka <i>Login</i>	147
5.5.2 Perancangan Antarmuka <i>Dashboard</i>	153
5.5.3 Perancangan Antarmuka <i>Users</i>	167
5.5.4 Perancangan Antarmuka <i>Ticket</i>	186
BAB VI.....	223
KESIMPULAN	223
6.1 Kesimpulan.....	223

DAFTAR GAMBAR

Gambar 1. 1 Karakteristik Sistem.....	3
Gambar 1. 2 Elemen Sistem.....	7
Gambar 1. 3 Contoh Aplikasi	25
Gambar 1. 4 Metode <i>Waterfall</i>	31
Gambar 1. 5 Proses dalam sistem <i>Monitoring</i>	35
Gambar 2. 1 <i>Diagram Metodelogi Penelitian</i>	55
Gambar 3. 1 Logo <i>Sublime</i>	58
Gambar 3. 2 <i>Contoh Tampilan Sublime Text</i>	59
Gambar 3. 3 Logo <i>XAMPP</i>	61
Gambar 3. 4 Contoh <i>Htdocs</i>	64
Gambar 3. 5 Contoh <i>PHPMyAdmin</i>	65
Gambar 3. 6 Contoh <i>Control Panel</i>	66
Gambar 3. 7 Ilustrasi <i>E-mail</i>	68
Gambar 3. 8 Arsitektur <i>E-mail Client/Server</i>	75
Gambar 3. 9 Layer <i>OSI</i>	81
Gambar 3. 10 Model STMP.....	81
Gambar 3. 11 Koneksi <i>Client – Server IMAP</i>	84
Gambar 3. 12 Contoh situs <i>http</i> : <i>Browser Google Chrome</i> menampilkan <i>status Website</i> dengan <i>protokol HTTP</i>	87
Gambar 3. 13 Status “ <i>Not Secure</i> ” pada <i>Browser Google Chrome</i>	88
Gambar 3. 14 Contoh situs <i>https</i> : Status “ <i>Secure</i> ” pada <i>Browser Google</i> <i>Chrome</i>	89
Gambar 4. 1 <i>Website Sublime</i>	101
Gambar 4. 2 Letak <i>Directory File .exe</i> Tersimpan	102
Gambar 4. 3 Proses Instalasi <i>Sublime</i>	102

Gambar 4. 4 Proses Instalasi <i>Sublime</i>	103
Gambar 4. 5 Proses Instalasi <i>Sublime</i>	103
Gambar 4. 6 Proses Instalasi <i>Sublime</i>	104
Gambar 4. 7 <i>Website XAMPP</i>	105
Gambar 4. 8 <i>Letak Directory</i>	106
Gambar 4. 9 <i>Proses Instalasi XAMPP</i>	106
Gambar 4. 10 Proses <i>Instalasi XAMPP</i>	107
Gambar 4. 11 <i>Proses Instalasi XAMPP</i>	108
Gambar 4. 12 Proses <i>Instalasi XAMPP</i>	109
Gambar 4. 13 Proses <i>Instalasi XAMPP</i>	110
Gambar 4. 14 Proses <i>Instalasi XAMPP</i>	111
Gambar 4. 15 Proses <i>Instalasi XAMPP</i>	112
Gambar 4. 16 Proses <i>Instalasi XAMPP</i>	113
Gambar 4. 17 <i>Tampilan XAMPP</i>	114
Gambar 4. 18 Hasil Akhir <i>XAMPP</i>	115
Gambar 4. 19 <i>Website PHP</i>	116
Gambar 4. 20 <i>System Properties</i>	118
Gambar 4. 21 <i>Edit System Variabel</i>	119
Gambar 4. 22 Menambahkan <i>Source Code</i>	120
Gambar 4. 23 Menambahkan <i>Source Code</i>	121
Gambar 4. 24 Menambahkan <i>Source Code</i>	121
Gambar 4. 25 Test <i>PHP</i>	122
Gambar 5. 1 <i>Bizagi modeler</i> sistem berjalan pada Sistem <i>Monitoring evaluasi kantor pusat ke kantor wiliyah</i>	124
Gambar 5. 2 <i>Bizagi modeler</i> sistem yang akan dibangun pada Sistem <i>Monitoring Job Desk OHC</i>	126
Gambar 5. 3 <i>Bizagi modeler</i> sub proses <i>Login</i> sistem yang akan dibangun.	
.....	127

Gambar 5. 4 <i>Bizagi modeler</i> sub proses penyelesaian tugas.	128
Gambar 5. 5 <i>Bizagi modeler</i> sub proses pembuatan tugas yang akan dibangun.....	130
Gambar 5. 6 <i>Bizagi modeler</i> sub proses konfirmasi tugas yang akan dibangun.....	130
Gambar 5. 7 <i>ConText Diagram</i> Sistem <i>Monitoring Job Desk OHC</i>	132
Gambar 5. 8 <i>Data Flow Diagram</i> Level 2 Aplikasi Sistem <i>Monitoring</i> Pada <i>Job Desk Human Capital</i>	133
Gambar 5. 9 <i>Data Flow Diagram</i> Level 2 Proses 1 <i>Kelola Data Users</i>	135
Gambar 5. 10 <i>Data Flow Diagram</i> Level 2 Proses 2 <i>Kelola Data Ticket</i> ...	136
Gambar 5. 11 <i>Data Flow Diagram</i> Level 2 Proses 3 <i>Kelola Data Department</i>	137
Gambar 5. 12 Struktur <i>Menu</i> Sistem <i>Monitoring</i> Terhadap <i>Job Desk Human Capital</i>	139
Gambar 5. 13 <i>Conceptual Data Model</i> Sistem <i>Monitoring</i> Terhadap <i>Job desc Human Capital</i>)	140
Gambar 5. 14 <i>Physical Data Model</i> Model Sistem <i>Monitoring</i> Terhadap <i>Job desc Human Capital</i>	141
Gambar 5. 15 <i>Database db_ocb</i>	143
Gambar 5. 16 Tabel <i>Department</i>	143
Gambar 5. 17 Tabel <i>ticket</i>	144
Gambar 5. 18 Tabel “ <i>ticket_activity</i> ”.....	145
Gambar 5. 19 Tabel <i>Users</i>	146
Gambar 5. 20 Antarmuka <i>Login Admin</i>	147
Gambar 5. 21 Antarmuka <i>Login Users</i>	148
Gambar 5. 22 Antarmuka <i>Dashboard Admin</i>	153
Gambar 5. 23 Antarmuka <i>Dashboard Users</i>	160
Gambar 5. 24 Antarmuka <i>Data Users</i>	167

Gambar 5. 25 Antarmuka <i>Add Data Users</i>	171
Gambar 5. 26 Antarmuka <i>Update Data Users</i>	175
Gambar 5. 27 Antarmuka <i>Change Password Users</i>	180
Gambar 5. 28 Perancangan Antarmuka <i>Delete Password Users</i>	182
Gambar 5. 29 Antarmuka <i>View Data Ticket</i>	186
Gambar 5. 30 Antarmuka <i>Add Data Ticket</i>	190
Gambar 5. 31 Antarmuka <i>Edit Data Ticket</i>	193
Gambar 5. 32 Perancangan Antarmuka <i>Delete data Ticket</i>	197
Gambar 5. 33 Antarmuka <i>View Data Department</i>	212
Gambar 5. 34 Antarmuka <i>Add Data Department</i>	216
Gambar 5. 35 Antarmuka <i>Edit Data Department</i>	218
Gambar 5. 36 Antarmuka <i>Delete Data Department</i>	220

DAFTAR TABEL

Tabel 1. 1 Dimensi dan Indikator <i>Job Description</i>	48
Tabel 3. 1 Perbandingan <i>E-mail</i> dengan surat biasa	69
Tabel 3. 2 <i>Field header pada E-mail</i>	70
Tabel 3. 3 <i>Header tambahan MIME</i>	78
Tabel 3. 4 Konten <i>Type</i>	79
Tabel 5. 1 Hasil simulasi proses sistem berjalan pada pada Sistem <i>Monitoring</i> evaluasi kantor pusat ke kantor wiliyah.....	124
Tabel 5. 2 Dokumen Monev Kantor Wilayah.....	125
Tabel 5. 3 Hasil simulasi proses sistem yang akan dibangun pada Sistem <i>Monitoring Job Desk OHC</i>	126
Tabel 5. 4 Hasil simulasi sub proses <i>Login</i> sistem yang akan dibangun..	128
Tabel 5. 5 Tabel simulasi sub proses penyelesaian tugas yang akan dibangun	
.....	129
Tabel 5. 6 Tabel simulasi sub proses pembuatan tugas yang akan dibangun	
.....	130
Tabel 5. 7 Tabel simulasi sub proses Konfirmasi tugas yang akan dibangun.	
.....	131
Tabel 5. 8 Data in, Data out <i>conText diagram</i>	132
Tabel 5. 9 Keterangan Data <i>Flow Diagram</i> Level 2 aplikasi Sistem <i>Monitoring</i> Pada <i>Job Desk Human Capital</i> (Proses keseluruhan).....	134
Tabel 5. 10 Keterangan Data <i>Flow Diagram</i> Level 2 Proses 1 Kelola Data <i>Users</i>	135
Tabel 5. 11 Keterangan Data <i>Flow Diagram</i> Level 2 Proses 2 Kelola Data <i>Ticket</i>	137

Tabel 5. 12 Keterangan <i>Data Flow Diagram</i> Level 2 Proses 3 Kelola Data <i>Department</i>	138
Tabel 5. 13 Kamus Alur Data	138

PENGANTAR

Dengan nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, Penulis panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, taufiq, serta hidayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan buku “**MEMBANGUN APLIKASI SISTEM MONITORING TERHADAP JOB DESK OPERATIONAL HUMAN CAPITAL (OHC)**” ini. Penulis menyadari sepenuhnya masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasa. Oleh karena itu, dengan tangan terbuka penulis menerima segala saran dan kritik dari pembaca agar menjadi lebih baik untuk kedepannya. Akhir kata penulis berharap buku ini dapat memberikan manfaat maupun inspirasi terhadap pembaca.

BAB I

DEFINISI

Sistem dan aplikasi adalah beberapa istilah yang digunakan dalam penyusunan buku ini. Penulis akan memaparkan penjelasan mengenai ketiga istilah tersebut.

1.1 Sistem

Sistem adalah suatu jaringan dari prosedur-prosedur yang saling berhubungan, berkumpul dan bersama-sama untuk menyelesaikan suatu sasaran tertentu. Kata Sistem berasal dari bahasa Latin (*systēma*) dan bahasa Yunani (*sustēma*). Dapat diartikan bahwa sistem adalah kesatuan bagian-bagian yang saling berhubungan yang berada dalam suatu wilayah serta memiliki penggerak, contoh umum misalnya seperti negara.

1.1.1 Penggunaan Istilah

Penggunaan istilah sistem itu meliputi diantaranya, yaitu:

1. Sistem yang digunakan untuk *Menunjukkan* suatu kumpulan yang disatukan oleh suatu bentuk saling hubungan yang saling ketergantungan yang teratur secara alamiah maupun oleh budi daya manusia sehingga menjadi suatu kesatuan yang hakiki, suatu yang berfungsi bersama-sama mengikuti suatu kontrol tertentu. Salah satu contohnya adalah ekosistem dan sistem tata surya.
2. Sistem yang berhubungan dengan alat-alat atau organ tubuh secara keseluruhan yang secara khusus memiliki hubungan terhadap fungsi organ tubuh tertentu yang rumit namun tetapi sangat vital. Salah satu contohnya adalah sistem saraf.
3. Sistem yang menunjukkan sekumpulan gagasan, prinsip serta doktrin hukum yang membentuk suatu hubungan yang dikenal sebagai isi buah

pemikiran tentang filsafat, agama atau pemerintahan tertentu. Salah satu contohnya adalah sistem pemerintahan demokratis.

4. Sistem yang digunakan untuk *Menunjukkan* suatu teori atau hipotesis. Salah satu contohnya adalah sistem pendidikan sistematik.
5. Sistem yang digunakan dalam tata acara atau metode. Salah satu contohnya adalah sistem mengetik dengan menggunakan sepuluh jari, sistem modul pembinaan atau pengajaran.
6. Sistem yang digunakan untuk *Menunjukkan* metode pengaturan organisasi atau susunan tata cara dalam metode pengelompokan, dan sebagainya. Salah satu contohnya adalah sistem pengelompokan bahan pustaka.

1.1.2 Pemakaian Sistem

Pemakaian sistem dapat digolongkan menjadi 2 secara garis besarnya, yaitu diantaranya:

1. Sistem sebagai suatu wujud entitas.

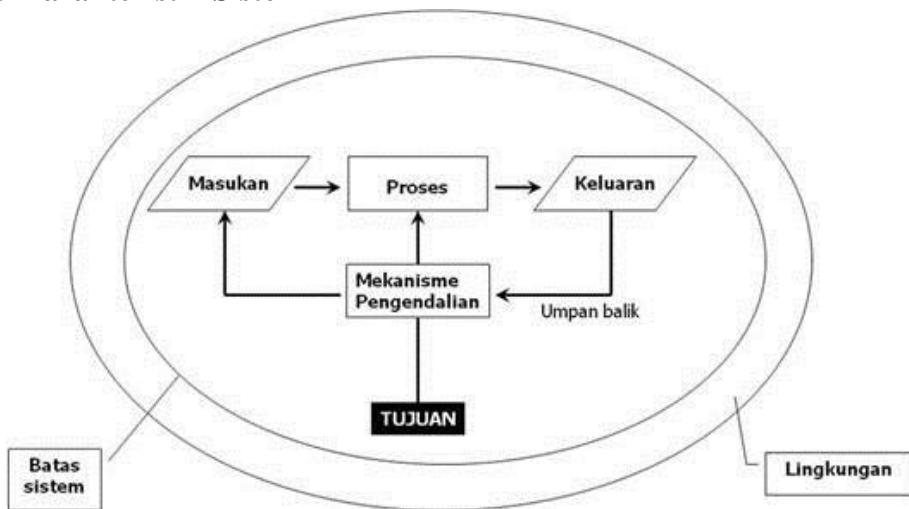
Suatu sistem dianggap suatu bagian yang saling berkaitan membentuk keseluruhan yang komplek. Sistem sendiri merupakan suatu wujud atau entitas sebagai suatu benda yang pada dasarnya bersifat deskriptif. Hal ini memberikan kemungkinan untuk membedakan atau menggambarkan antara benda yang satu dengan yang lainnya untuk kepentingan penganalisaan untuk mempermudah pemecahan masalah.

2. Sistem sebagai suatu metode.

Sistem mempunyai makna metodologi, sistem yang digunakan akan *Menunjukkan* tata cara atau prosedur yang bersifat preskriptif bukan deskriptif. Salah satu contohnya adalah deskriptif (program investasi) dan preskriptif (program investasi yang meningkatkan dividen). Contoh deskriptif *Menunjukkan* wujud barang dan preskriptif *Menunjukkan* suatu

metode atau cara untuk mencapai sesuatu. Konsep ini dikenal dalam pengertian sebagai pendekatan sistem. Pendekatan tersebut adalah penerapan metode ilmiah dalam usaha memecahkan masalah. Menentukan pemahaman bahwa setiap sistem berada dari sistem yang besar atau luas sehingga semua benda dengan suatu cara yang saling berkaitan.

1.1.3 Karakteristik Sistem



Gambar 1. 1 Karakteristik Sistem

Karakteristik sistem diantaranya meliputi sebagai berikut:

1. Memiliki komponen.

Suatu sistem terdiri dari sejumlah komponen yang saling berhubungan atau berinteraksi, bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem yang dapat berupa suatu subsistem atau bagian dari sistem. Setiap sistem tidak peduli seberapa pun kecilnya selalu mengandung komponen atau subsistem. Setiap subsistem mempunyai sifat dari sistem untuk menjalankan fungsi tertentu dan mempengaruhi cara

kerja sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang biasa disebut sebagai supra sistem. Salah satu contohnya adalah perusahaan dapat disebut dengan suatu sistem dan industri yang merupakan sistem yang lebih besar sehingga dapat disebut dengan supra sistem. Terlihat dari pandangan industri sebagai suatu sistem, maka perusahaan dapat disebut sebagai subsistem. Jika perusahaan dipandang sebagai suatu sistem, maka sistem akuntansi adalah subsistemnya.

2. Batas sistem (*boundary*).

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan di luarnya. Batas sistem ini memungkinkan suatu sistem yang dipandang sebagai suatu kesatuan. Batasan suatu sistem untuk *Menunjukkan* ruang lingkup dari sistem tersebut.

3. Lingkungan luar sistem (*environment*).

Lingkungan luar sistem merupakan apapun yang berada diluar batas dari sistem yang mempengaruhi operasi sistem.

4. Penghubung sistem (*interface*).

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem lainnya.

5. Masukan sistem (*input*).

Masukan sistem merupakan energi yang dimasukkan kedalam sistem. Masukkan dapat berupa perawatan (*maintenance input*) dan masukkan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan

supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk mendapatkan keluaran. Salah satu contohnya adalah sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputer dan data adalah sebagai signal input untuk diolah menjadi informasi.

1.1.4 Klasifikasi Sistem

Klasifikasi sistem diantaranya meliputi sebagai berikut:

1. Sistem abstrak merupakan sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Salah satu contohnya adalah sistem teologi.
2. Sistem fisik merupakan sistem yang ada secara fisik. Salah satu contohnya adalah sistem komputer, sistem akuntansi, sistem produksi dll.
3. Sistem alamiah merupakan sistem yang terjadi melalui proses alam. Salah satu contohnya adalah sistem matahari, sistem luar angkasa, sistem reproduksi dll.
4. Sistem buatan manusia merupakan sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi manusia dengan mesin disebut *human machine system*. Salah satu contohnya adalah contoh sistem informasi.
5. Sistem tertentu (*deterministic system*) merupakan beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi bagian-bagiannya dapat dideteksi dengan pasti sehingga keluaran dari sistem dapat diramalkan. Salah satu contohnya adalah sistem komputer.
6. Sistem tak tentu (*probabilistic system*) merupakan sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

7. Sistem tertutup (*close system*) merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan sistem luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak luarnya. Secara teoritis sistem tersebut ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah *relatively closed system*.
8. Sistem terbuka (*open system*) merupakan sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya.

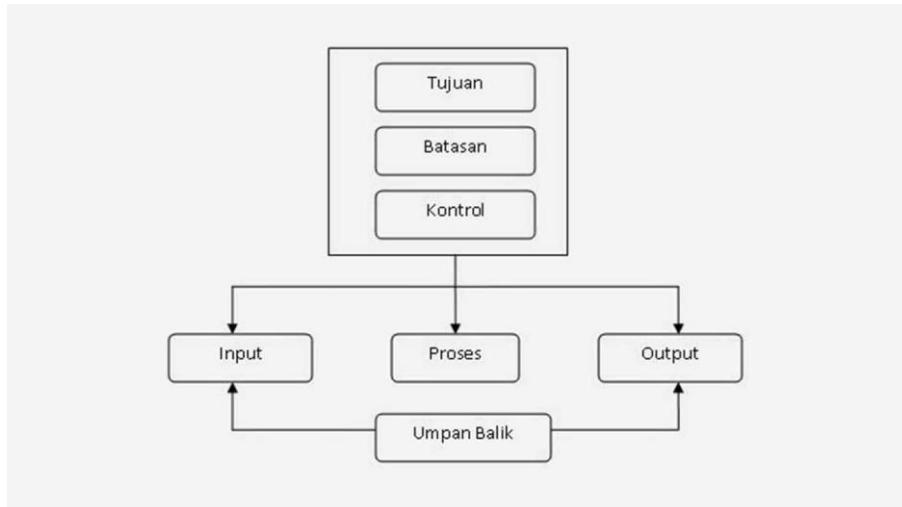
Lebih spesifik dikenal juga yang disebut dengan sistem terotomasi yang merupakan bagian dari sistem buatan manusia dan berinteraksi dengan kontrol oleh satu atau lebih komputer sebagai bagian dari sistem yang digunakan dalam masyarakat modern.

1.1.5 Jenis Sistem

Ada berbagai tipe sistem berdasarkan kategori:

1. Atas dasar keterbukaan:
 - a. Sistem terbuka, di mana pihak luar dapat mempengaruhinya.
 - b. Sistem tertutup.
2. Atas dasar komponen:
 - a. Sistem fisik, dengan komponen materi dan energi.
 - b. Sistem non-fisik atau konsep, berisikan ide-ide.

1.1.6 Elemen Sistem



Gambar 1. 2 Elemen Sistem

Ada beberapa elemen yang membentuk sebuah sistem, berikut penjelasannya sebagai berikut:

1. Tujuan

Setiap sistem memiliki tujuan, entah hanya satu atau lebih. Tujuan inilah yang memotivasi dan mengarahkan sistem. Tanpa tujuan, sistem menjadi tidak terarah dan tak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem yang lain berbeda.

2. Masukan

Masukan (*input*) sistem merupakan segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan untuk diproses. Masukan bisa berupa hal yang berwujud (tampak secara fisik) maupun yang tidak tampak. Contoh masukan yang berwujud adalah bahan mentah, sedangkan contoh yang tidak berwujud adalah informasi (misalnya permintaan jasa pelanggan).

3. Proses

Proses adalah bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna dan lebih bernilai, misalnya berupa informasi dan produk, tetapi juga dapat berupa hal yang tidak berguna, misalnya saja sisa pembuangan atau limbah. Pada pabrik kimia, proses dapat berupa bahan mentah. Pada rumah sakit, proses dapat berupa aktivitas pembedahan pasien.

4. Keluaran

Keluaran (*output*) adalah hasil dari proses. Pada sistem informasi berupa informasi atau laporan, dan lain-lain.

5. Batas

Yang disebut batas sistem adalah pemisah antara sistem dan daerah di luar sistem (lingkungan). Batas sistem menentukan konfigurasi, ruang lingkup, atau kemampuan sistem. Tentu saja batas sebuah sistem dapat dikurangi atau dimodifikasi sehingga akan mengubah perilaku sistem.

6. Mekanisme Pengendalian dan Umpaman Balik

Mekanisme pengendalian diwujudkan dengan menggunakan umpan balik, yang mencuplik keluaran. Umpaman balik ini digunakan untuk mengontrol baik masukan maupun proses.

7. Lingkungan

Lingkungan bisa berpengaruh terhadap operasi sistem, dalam arti bisa merugikan atau menguntungkan sistem itu sendiri.

1.1.7 Sistem Berdasarkan Prinsip

Sistem berdasarkan prinsip dasar secara umum terbagi dalam beberapa macam, diantaranya yaitu:

1. Sistem terspesialisasi adalah sistem yang sulit diterapkan pada lingkungan yang berbeda, misalnya sistem biologi, ikan yang dipindahkan ke darat.
2. Sistem besar adalah sistem yang sebagian besar sumber dayanya berfungsi melakukan perawatan harian, misalnya dinosaurus sebagai sistem biologi menghabiskan sebagian besar masa hidupnya dengan makan dan makan.
3. Sistem sebagai bagian dari sistem lain adalah sistem selalu merupakan bagian dari sistem yang lebih besar, dan dapat terbagi menjadi sistem yang lebih kecil.
4. Sistem berkembang merupakan sistem yang walaupun tidak berlaku bagi semua sistem tetapi hampir semua sistem selalu berkembang.

1.1.8 Pelaku Sistem

Pelaku sistem terdiri dari 6 kelompok, diantaranya sebagai berikut:

1. Pemakai, pada umumnya ada 3 jenis pemakai, yaitu operasional, pengawas dan eksekutif.

2. Pemeriksa, ukuran dan kerumitan sistem yang dikerjakan dan bentuk alami organisasi dimana sistem tersebut diimplementasikan dapat menentukan kesimpulan perlu tidaknya pemeriksa. Pemeriksa biasanya menentukan segala sesuatunya berdasarkan ukuran-ukuran standar yang dikembangkan pada banyak perusahaan sejenis.
3. Penganalisa sistem, fungsi-fungsinya antara lain yaitu:
 - a. Arkeolog yaitu yang menelusuri bagaimana sebenarnya sistem lama berjalan, bagaimana sistem tersebut dijalankan dan segala hal yang menyangkut sistem lama.
 - b. Inovator, yaitu yang membantu mengembangkan dan membuka wawasan pemakai bagi kemungkinan-kemungkinan lain.
 - c. Mediator, yaitu yang menjalankan fungsi komunikasi dari semua level, antara lain: pemakai, manajer, programmer, pemeriksa dan pelaku sistem yang lainnya yang mungkin belum punya sikap dan cara pandang yang sama.
 - d. Pimpinan proyek, penganalisa sistem haruslah personil yang lebih berpengalaman dari programmer atau desainer. Selain itu mengingat penganalisa sistem umumnya ditetapkan terlebih dahulu dalam suatu pekerjaan sebelum yang lain bekerja, adalah hal yang wajar jika penanggung jawab pekerjaan menjadi porsi penganalisa sistem.
4. Pendesain sistem merupakan sistem menerima hasil penganalisa sistem berupa kebutuhan pemakai yang tidak berorientasi pada teknologi tertentu, yang kemudian ditransformasikan ke desain arsitektur tingkat tinggi dan dapat diformulasikan oleh programmer.

5. Programmer mengerjakan dalam bentuk program dari hasil desain yang telah diterima dari pendesain.
6. Personil pengoperasian bertugas dan bertanggung jawab di pusat komputer misalnya jaringan, keamanan perangkat keras, keamanan perangkat lunak, pencetakan dan *backup*. Pelaku ini mungkin tidak diperlukan bila sistem yang berjalan tidak besar dan tidak membutuhkan klasifikasi khusus untuk menjalankan sistem.

1.1.9 Analisis Sistem

Penganalisa sistem merupakan bagian dari tim yang berfungsi mengembangkan sistem yang memiliki daya guna tinggi dan memenuhi kebutuhan pemakai akhir. Pengembangan ini dipengaruhi sejumlah hal, yaitu:

1. Produktivitas, saat ini dibutuhkan sistem yang lebih banyak, lebih bagus dan lebih cepat. Hal ini membutuhkan lebih banyak programmer dalam penganalisa sistem yang berkualitas, kondisi kerja ekstra, kemampuan pemakai untuk mengambil sendiri, bahasa pemrograman yang lebih baik, perawatan sistem yang lebih baik (umumnya 50 % sampai 70 % sumber daya digunakan untuk perawatan sistem), disiplin teknis pemakaian perangkat lunak dan perangkat pengembangan sistem yang terotomasi.
2. Reliabilitas, waktu yang dihabiskan untuk testing sistem secara umum menghabiskan 50% dari waktu total pengembangan sistem. Dalam kurun waktu 30 tahun sejumlah sistem yang digunakan di berbagai perusahaan mengalami kesalahan dan ironisnya sangat tidak mudah untuk mengubahnya. Jika terjadi kesalahan, ada dua cara yang bisa

dilakukan, yaitu melakukan pelacakan sumber kesalahan dan harus menemukan cara untuk mengoreksi kesalahan tersebut dengan mengganti program, menghilangkan sejumlah statement lama atau menambahkan sejumlah statement baru.

3. Maintabilitas, perawatan mencakup:

- a. Modifikasi sistem sesuai perkembangan perangkat keras untuk meningkatkan kecepatan pemrosesan (yang memegang peranan penting dalam pengoperasian sistem)
- b. Modifikasi sistem sesuai perkembangan kebutuhan pemakai. Antara 50% sampai 80% pekerjaan yang dilakukan pada kebanyakan pengembangan sistem dilakukan untuk revisi, modifikasi, konversi, peningkatan dan pelacakan kesalahan.

1.1.10 Sistem Terotomasi

Sistem terotomasi mempunyai sejumlah komponen yaitu diantaranya:

1. Perangkat keras, misalnya CPU, *disk*, *printer*, *tape*.
2. Perangkat lunak, misalnya sistem operasi, sistem *Database*, program pengontrol komunikasi, program aplikasi.
3. Personil yang mengoperasikan sistem, menyediakan masukan, mengkonsumsi keluaran dan melakukan aktivitas manual yang mendukung sistem.
4. Data yang harus tersimpan dalam sistem selama jangka waktu tertentu)

5. Prosedur instruksi dan kebijakan untuk mengoperasikan sistem.

Sistem terotomasi terbagi dalam sejumlah kategori :

1. *On-line systems.*

Sistem *on-line* adalah sistem yang menerima langsung *input* pada area, dimana *input* tersebut direkam dan menghasilkan *output* yang dapat menghasilkan komputasi pada area. Area dapat dipisah-pisah dalam skala. Salah satu contohnya adalah ratusan kilometer. Biasanya digunakan bagi reservasi angkutan udara, reservasi kereta api, perbankan, dll.

2. *Real-time systems.*

Sistem *real-time* adalah mekanisme pengontrolan, pemrosesan, dan perekaman data yang sangat cepat sehingga *output* yang dihasilkan dapat diterima dalam waktu yang relatif sama. Perbedaannya dengan sistem *on-line* adalah satuan waktu yang digunakan *real-time* biasanya seperseratus atau seperseribu detik, sedangkan *on-line* masih dalam skala detik atau bahkan kadang memerlukan waktu beberapa menit. Perbedaan lainnya, *on-line* biasanya hanya berinteraksi dengan pemakai, sedangkan *real-time* berinteraksi langsung dengan pemakai dan lingkungan yang dipetakan.

3. *Decision support System + strategic planning system.*

Sistem yang memproses transaksi organisasi secara harian dan membantu para manajer mengambil keputusan, mengevaluasi dan menganalisis tujuan organisasi. Digunakan untuk sistem penggajian, sistem pemesanan, sistem akuntansi dan sistem produksi. Biasanya berbentuk paket statistik, paket pemasaran, dll. Sistem ini tidak hanya merekam dan menampilkan data tetapi juga fungsi matematik, data analisa statistik dan

menampilkan informasi dalam bentuk grafik, tabel, chart sebagaimana laporan konvensional.

4. *Knowledge-based system.*

Program komputer yang dibuat mendekati kemampuan dan pengetahuan. Umumnya menggunakan perangkat keras dan perangkat lunak seperti PROLOG dan LISP.

1.1.11 Pengantar Aplikasi

Aplikasi adalah sebuah program siap pakai yang bisa dipakai untuk menjalankan sejumlah perintah dari pengguna aplikasi itu sendiri. Dengan tujuan untuk memperoleh hasil yang lebih akurat dan sesuai dengan tujuan pembuatan aplikasi tersebut. Aplikasi juga memiliki pengertian sebagai pemecah masalah yang memakai salah satu teknik pemrosesan data aplikasi yang mengacu pada sebuah komputerisasi atau smartphone yang diinginkan atau diharapkan. Aplikasi berasal dari kata bahasa Inggris “*Application*” yang artinya merupakan bentuk dari kata kerja *to apply* atau dalam bahasa Indonesia artinya pengolah. Secara istilah, aplikasi komputer adalah subkelas perangkat lunak komputer yang memakai kemampuan komputer dengan langsung melaksanakan suatu tugas yang diinginkan pengguna tersebut. Selain itu, pengertian aplikasi adalah suatu perintah yang mengeksekusi dalam memberikan sebuah petunjuk kerja serta fungsi yang diinginkan. Dan pengertian aplikasi secara umum adalah sebagai alat terapan yang berfungsi secara khusus serta terpadu sesuai kemampuan yang dipunyai aplikasi adalah suatu perangkat komputer yang sudah siap dipakai sebagai *Users*.

Kesimpulan dari pengertian diatas adalah bahwa aplikasi ini merupakan *software* yang fungsinya untuk melaksanakan berbagai bentuk pekerjaan

maupun tugas tertentu misalnya seperti penerapan, pemakaian dan juga penambahan data.

1.1.12 Sejarah Aplikasi

Aplikasi yang kita temui saat ini sebenarnya mempunyai sejarah yang panjang. Dari awal kemunculannya, aplikasi merupakan sebuah program sederhana hingga pada akhirnya berubah menjadi sesuatu yang sangat dibutuhkan pada saat ini. Aplikasi bersumber dari *Aljabar Booelan*, yang memakai kode *binary* digit (bit) yang terdiri dari 2 angka, yaitu *on* yang berarti benar dan *off* yang berarti salah. Pemakaian kode *binary* digit ini membuat masyarakat pada saat itu menyusun beberapa kelompok bit yang terdiri dari 4 bit (*nible*), 8 bit (*byte*), 2 *byte* (*word*) dan 32 bit (*double bit*).

Pengelompokan tersebut membantu *software* komputer pada bermacam kegiatan. Seperti merakit beberapa kode menjadi struktur instruksi. Struktur instruksinya seperti operasi logika, penyimpanan, *transfer* hingga membentuk kode baru yang disebut sebagai *Assembler*. Kode itulah yang menjadi cikal bakal untuk membuat berbagai jenis aplikasi yang bisa dipakai untuk mempermudah kegiatan manusia.

Di dalam sejarahnya, aplikasi mengalami beberapa evolusi. Pertama adalah Era Pioner. Pada awalnya perangkat lunak terdiri dari sambungan kabel antar bagian di komputer, cara lain untuk mengakses komputer dengan *punched card* atau kartu yang dilubangi. Pemakaian komputer dilakukan dalam suatu program dengan sebuah mesin dan tujuan tertentu secara langsung. Pada masa tersebut perangkat lunak menjadi satu kesatuan dengan perangkat kerasnya. Yang dihasilkan dari penggunaan komputer yang dilakukan secara langsung adalah *print out*. Proses yang dilakukan komputer terdiri dari sejumlah baris instruksi yang berurutan.

Era selanjutnya adalah Era Stabil, dimana penggunaan komputer berkembang lebih pesat, khususnya pada perusahaan dan industri. Beberapa baris perintah dijalankan secara multitasking atau serempak. Dalam era ini dikenal dengan sistem basis data yang menjadi pemisah antara data dengan program. Pada era ini aplikasi komputer sudah mengalami kemajuan yang cukup pesat. Baris-baris perintah aplikasi komputer yang dioperasikan oleh komputer tidak lagi satu per satu, namun sudah bisa melakukan banyak proses secara bersamaan atau *multitasking*. Aplikasi pada era stabil juga dapat menyelesaikan banyak pengguna atau *multi Users* dan cepat atau langsung (*real time*).

Setelah Era Stabil ada Era Mikro. Dalam era ini perangkat lunak dibedakan menjadi beberapa macam. seperti aplikasi sistem (*Windows*, *Machintosh*, *Linux* dan lain-lain), software aplikasi (*Ms. Office*, *Corel*, *Adobe*, dan sebagainya), serta bahasa pemrograman atau *language software* (*Assembler*, *Delphi*, *Visual Basic* dan lain-lain). Era terakhir adalah Era Modern. Di mana pada era ini tidak hanya terciptanya suatu *supercomputer* yang memiliki 25 prosesor saja, namun juga perangkat lunak yang tersebar merata di berbagai tempat. Kini PC dapat disinkronkan dengan perangkat genggam. Beberapa aplikasi juga bisa dioperasikan di ponsel dengan sistem operasi *Android*, *iOS* dan lain-lain. Di Era Modern perkembangan aplikasi komputer semakin pesat dan juga bisa diaplikasikan ke dalam perangkat lain. Kemampuan pada aplikasi juga mengalami peningkatan. Selain menangani masalah teknis aplikasi juga sudah bisa mengenal suara dan juga gambar.

1.1.13 Jenis-Jenis Aplikasi Komputer

Jenis-jenis aplikasi komputer yang ada atau telah dipergunakan, yaitu:

1. Aplikasi grafis.

Aplikasi grafis merupakan suatu program untuk mengolah data yang berformat gambar baik dengan membuat gambar baru maupun mengubah gambar yang sudah dibuat sebelumnya. *Software* aplikasi grafis dibagi menjadi tiga macam, yaitu:

a. Aplikasi grafis berbasis *vector*

Contoh: *Adobe Illustrator, Corel Draw, Macromedia Freehand, Micrografx designe.*

b. Aplikasi grafis berbasis *pixel/BITMAP*

Contoh: *Adobe Photoshop, Macromedia Fireworks, Corel Photopaint.*

c. Aplikasi grafis berbasis tata letak

Contoh: *Adobe FrameMaker, Adobe In Design, Adobe PageMaker, Corel Ventura, Microsoft Publisher, Quark Xpress.*

2. Aplikasi *Web Browser*

Merupakan bagian dari internet sebagai komunitas jaringan komputer yang memberikan pelayanan http (*world wide web*). Dengan demikian, definisi teknis dari *world wide web* adalah semua sumber daya dan semua pengguna di internet yang menggunakan HTTP (*HyperText Transfer Protocol*). WWW adalah aplikasi yang paling menarik di Internet dan seperti email, aplikasi ini sangat penting dan banyak digunakan. Aplikasi *web Browser* yang sering digunakan antara lain, *Opera, Internet Explorer, Safari, Firefox* dan *Chrome*.

3. Animasi

Merupakan bentuk seni yang tampak secara spontan menimbulkan gerakan kehidupan pada suatu objek. Sebenarnya terdapat beberapa fungsi yang berbeda untuk menghasilkan animasi berbasis komputer dan satu daripadanya ialah animasi tiga dimensi (3D). Fungsi lain untuk mencipta animasi komputer ialah dengan menggunakan alat pengecatan komputer yang standar untuk mengecat frame-frame tunggal sebelum dilakukan proses penggabungan. Ini kemudian disimpan sebagai sebuah *File* gambar (*movie*).

4. Aplikasi Pendidikan

a. *Computer Assisted Instruction* (CAI)

Komputer secara langsung digunakan dalam proses belajar, sebagai pengganti pengajar ataupun buku. Beberapa aplikasi CAI antara lain, *Drill and Practice*, Tutorial, Simulasi.

b. *Computer Managed Instruction* (CMI)

Para pengajar memanfaatkan komputer untuk merencanakan kuliah, disesuaikan dengan kondisi para siswa, yang terdiri dari acara belajar dengan bantuan komputer, membaca, dan ujian.

c. *Computer Assisted Testing* (CAT)

Komputer digunakan sebagai media ujian untuk menggali kemampuan siswa dengan cara-cara tanya jawab secara aktif. Bentuknya bermacam-macam, dari mulai yang sederhana dimana komputer (biasanya melalui layar peraga) hingga bentuk yang lebih maju.

5. Aplikasi Multimedia

Multimedia merupakan *software* yang digunakan untuk menghubungkan komputer dengan peralatan multimedia seperti kamera

video, kamera digital, video. Salah satu contohnya adalah *Windows Media Player, Winamp, PowerDVD, Klite, VLC Media Player*.

6. Antivirus

Antivirus merupakan program yang digunakan untuk mendeteksi dan menghilangkan virus yang tertular pada komputer yang sedang dipakai. Salah satu contohnya adalah *McAfee VirusScan, Norton Antivirus, AVG*.

7. Aplikasi Komunikasi

Aplikasi yang satu ini merupakan yang saat ini paling banyak digunakan dan merupakan yang paling populer. Aplikasi ini digunakan agar manusia bisa berkomunikasi dengan pengguna komputer, *smartphone* atau gadget lain. Salah satu contohnya adalah *DataFax, Carbon Copy, CrossTalk, Line, BBM, Whatsapp*, dll.

8. Aplikasi DBMS

Aplikasi DBMS (*Database Management System*) digunakan untuk menyimpan data, mengolah data, serta menghasilkan *output* berupa informasi. Aplikasi seperti ini ada yang tersedia secara gratis, namun ada juga yang berbayar. Salah satu contohnya adalah *MySQL, Microsoft Access, Oracle, Foxpro*, dan lain-lain. Komponen dari sebuah DBMS adalah sebagai berikut :

1. *Query Processor*

Database Manager menghubungkan program aplikasi *Users-submitted* dan *query*. *Database Manager* menerima *query* dan memeriksa skema *eksternal* dan konseptual

untuk menentukan *record* konseptual apa yang diperlukan untuk memuaskan permintaan.

2. *Database Manager*

Database Manager menghubungkan program aplikasi *Users-submitted* dan *query*. *Database Manager* menerima *query* dan memeriksa skema *eksternal* dan konseptual untuk menentukan *record* konseptual apa yang diperlukan untuk memuaskan permintaan.

3. *File Manager*

File Manager memanipulasi penyimpanan *File* dan mengatur penempatan ruang penyimpanan dalam *disk*. Komponen ini mendirikan dan memelihara daftar struktur dan indeks yang didefinisikan dalam skema *internal*.

4. *DML Preprocessor*

Modul ini mengubah pernyataan DML yang tertanam dalam program aplikasi ke dalam pemanggilan fungsi *standard* dalam *host language*. Komponen ini harus berinteraksi dengan *query processor* untuk membuat kode yang sesuai.

5. *DDL Compiler*

Modul ini mengubah pernyataan DDL ke dalam seperangkat tabel berisi *metadata*. Tabel ini kemudian

disimpan dalam katalog sistem sementara itu informasi kendali disimpan dalam *handler File* data.

6. *Catalog Manager*

Mengatur akses dan memelihara katalog sistem. Katalog sistem diakses oleh sebagian besar komponen DBMS.

1.1.14 Klasifikasi Aplikasi

Demi mempermudah mengenal aplikasi, maka aplikasi biasanya diklasifikasikan menjadi 7 klasifikasi macam, yaitu:

1. *System software*, perangkat lunak inilah yang akan mengelola serta mengendalikan jalannya operasi internal pada sistem komputer.
2. *Real time software*, yaitu perangkat lunak yang bertugas menganalisa, mengamati, serta mengendalikan kejadian pada dunia nyata saat sedang terjadi.
3. *Business software*, yaitu perangkat lunak yang digunakan untuk mengatur sistem keuangan, dimana pengelolaan uang menjadi hal yang sangat fundamental bagi sebagian masyarakat.
4. *Engineering and scientific software*, perangkat lunak ini berfungsi untuk membantu dalam pengembangan teknologi dan dalam kegiatan penelitian.

5. *Artificial intelligence software*, yakni aplikasi yang digunakan untuk membantu memecahkan sebuah masalah yang bersifat non algoritmik, yang mana tidak sesuai dengan perhitungan dan analisis secara langsung.
6. *Web based software*, merupakan perangkat lunak yang berfungsi untuk mengantarkan pengguna pada akses internet secara langsung.
7. Personal *computer software*, perangkat lunak yang terakhir ini merupakan perangkat pengguna resmi dan juga pribadi, yang telah banyak digunakan sejak dua dekade terakhir ini.

1.1.15 Fungsi Aplikasi

Sebagai yang sudah disinggung di atas, aplikasi mempunyai beberapa fungsi. Salah satunya adalah untuk mempermudah kegiatan penggunanya. Secara spesifik aplikasi mempunyai beragam fungsi, baik di dunia pendidikan, industri dan manufaktur, bisnis dan perbankan, militer hingga kedokteran.

1. Fungsi Aplikasi dalam Pendidikan

Fungsi aplikasi dalam dunia pendidikan tentu saja untuk menambah pengetahuan. Selain itu aplikasi juga bisa bermanfaat untuk mempermudah proses belajar mengajar antara guru dengan siswanya.

2. Tenaga pengajar bisa memanfaatkan aplikasi sebagai bahan untuk pembelajaran. Contohnya mencari bahan untuk belajar, hingga informasi lain seperti melihat data nilai siswa, jadwal pembelajaran, beasiswa dan lain-lain. Dengan aplikasi, maka seluruh kegiatan pembelajaran bisa dilakukan dengan lebih mudah dan maksimal.

3. Industri dan Manufaktur

Penggunaan aplikasi pada komputer ini tentu saja dapat digunakan untuk mengkoordinasi mengenai penggunaan mesin dan mengontrolnya. Hal ini dilakukan agar hasil yang diberikan memiliki kualitas dan kuantitas yang lebih banyak. Dengan demikian akan mendapatkan kenyamanan yang diberikan. Untuk mendapatkan mengenai proses yang demikian tentu saja akan meningkatkan keuntungan yang diberikan. Sehingga akan meningkatkan hasil yang diinginkan dengan memberikan format khusus tanpa mengeluarkan banyak biaya yang besar.

4. Bisnis dan Perbankan

Didalam suatu bisnis, perlu menghitung besarnya keuntungan yang diperoleh. Jika keuntungan dihitung secara manual maka akan membutuhkan waktu yang lama. oleh karena itu diperlukannya aplikasi untuk menghitung besar keuntungan tersebut. Dalam menjalani bisnis tentu saja aplikasi komputer sangat digunakan untuk menghitung keuntungan, proses produksi dan masih banyak lainnya. Bahkan apabila dalam bidang bisnis tentu saja aplikasi ini akan memberikan lapangan pekerjaan baru yang menghasilkan pundi-pundi keuntungan. Sedangkan untuk perbankan tentu saja akan membantu dalam memudahkan perhitungan prosentase bunga, layanan keuangan dan masih banyak lainnya. Dengan demikian hasilnya menjadi semakin maksimal dirasakan untuk mendapatkan kenyamanan dalam mengatur segala kepentingan yang dimiliki.

5. Militer

Aplikasi juga membantu dalam kemampuan untuk mempertahankan negara dengan digunakannya untuk keperluan militer. Tentu saja akan memudahkan dalam mengembangkan dalam mengontrol pesawat, perhitungan ketinggian, mengontrol *navigasi*, peluru selam dan masih banyak lainnya. Dengan keamanan dan kecanggihan militer yang demikian. Tentu saja hasilnya yang didapatkan menjadi semakin optimal sesuai dengan keinginan kita. Usahakan dalam menggunakan dalam militer pastikan penggunaannya secara bijaksana. Tujuannya tentu saja agar hasilnya menjadi semakin bermanfaat untuk mendapatkan hasil yang sesuai dengan sasaran dan keinginan.

6. Kedokteran

Siapa sangka adanya aplikasi yang dikembangkan khusus untuk bidang kedokteran. Misalnya saja, adanya aplikasi untuk mendiagnosa penyakit, menawarkan perawatan rutin, bahkan sampai meracik obat. Dengan demikian, kita dapat mengecek diri sendiri melalui aplikasi tersebut. Hal ini karena dalam bidang kedokteran aplikasi ini akan digunakan dalam mendiagnosa sebuah penyakit, memilih obat yang dapat menyembuhkan dan meracik obat tersebut. Tentu saja dengan adanya bantuan yang demikian hasilnya menjadi semakin maksimal sesuai dengan keinginan kita. Mulailah menggunakan aplikasi komputer yang dimiliki tersebut sesuai dengan keinginan yang diinginkan.

1.1.16 Ciri – Ciri Kualitas Aplikasi

Sejumlah ciri-ciri yang menandakan kualitas pada sebuah aplikasi, yaitu:

1. Aplikasi bisa memenuhi kebutuhan *Users*.
2. Merespon sebuah instruksi secara cepat.

3. Bisa berjalan diberagam *platform*.
4. Bisa membutuhkan *resource* (prosesor dan media penyimpanan) yang rendah.

1.1.17 Contoh Aplikasi



Gambar 1. 3 Contoh Aplikasi

Berikut ini adalah beberapa contoh aplikasi yang banyak digunakan:

1. *Adobe Reader* yaitu aplikasi yang berfungsi untuk membaca dokumen, seperti .doc, .docx, dan .pdf.
2. *CCleaner* yaitu aplikasi yang berfungsi untuk membersihkan *junk File* atau *File* sampah dari perangkat komputer, *handphone* sehingga dapat meningkatkan kinerja perangkat tersebut.
3. *Game* yaitu aplikasi yang digunakan sebagai media hiburan untuk bermain *game*.

4. *Google Chrome, Mozilla Firefox* yaitu aplikasi yang berfungsi untuk menjelajahi internet.

5. *Instagram* yaitu aplikasi yang digunakan untuk membagikan foto atau video kepada orang lain.

6. *Internet Download Manager* yaitu aplikasi yang digunakan untuk mempercepat proses *download*.

7. *Microsoft Access, Paradox* yaitu aplikasi yang digunakan untuk menyimpan, mengolah data dalam jumlah yang besar.

8. *Microsoft Word* yaitu aplikasi yang digunakan untuk mengolah kata, seperti mengetik dokumen, membaca dokumen dan meng-*Edit* dokumen.

9. *Microsoft Excel* yaitu aplikasi ini digunakan untuk mengolah angka dan tabel untuk dihitung.

10. *Microsoft Powerpoint* yaitu aplikasi yang digunakan untuk membuat dan menampilkan data yang bersifat presentatif.

11. *Notepad* yaitu dahulu *notepad* berfungsi sebagai catatan kecil. Namun sekarang, *notepad* digunakan para pemrogram untuk *Menuliskan script* dan menyimpannya dengan berbagai ekstensi.

12. *Smadav, Avast* yaitu aplikasi yang digunakan untuk melindungi perangkat dari *malware* dan virus.
13. *Shareit* yaitu aplikasi yang dapat digunakan untuk melakukan transfer data ke perangkat lain.
14. *SPSS* yaitu aplikasi yang digunakan untuk mengolah data statistik.
15. *Adobe Photoshop, Corel Draw* yaitu aplikasi yang berfungsi untuk mengolah gambar.
16. *Sound Recorder* yaitu aplikasi yang digunakan untuk merekam suara.
17. *WhatsApp* yaitu aplikasi yang digunakan untuk berinteraksi dengan orang lain.
18. *Winrar* yaitu aplikasi yang digunakan untuk mengompres ukuran *File*.
19. *Winamp, GOM Player, Windows Media Player* yaitu aplikasi yang digunakan memutar lagu atau video dengan berbagai *format*.

1.2 Bahasa Pemrograman

1.2.1 Sejarah PHP

Sejarah awal *PHP* adalah pendekatan dari Personal Home Page (situs personal), dulu *PHP* masih berbentuk script yang berfungsi sebagai pengolah data form dari *web Server-side* yang memiliki sifat open source, dan memiliki nama Form Interpreter (FI). *PHP* (HyperText Preprocessor) memiliki sintak yang mirip dengan ASP, Java, bahasa C, Perl dan memiliki kelebihan fungsi yang mudah dipahami dan spesifik.

Kelebihan *PHP* yaitu:

1. Bahasa pemrograman *PHP* tidak memerlukan kompilasi atau Compiler dalam penggunaanya.
2. Memiliki sifat open source.
3. Banyak aplikasi *PHP* gratis dan siap untuk digunakan seperti PrestaShop, WordPress, dll.
4. Bisa membuat *web* menjadi dinamis
5. *PHP* memiliki banyak dukungan dari berbagai *web Server* contohnya saja Appache.
6. Mudahnya mengembangkan aplikasi *PHP*
7. *PHP* memiliki keunggulan lebih cepat dibandingkan dengan Java dan ASP.
8. MySQL merupakan paket aplikasi dengan *PHP*.

Kekurangan *PHP* yaitu:

1. *PHP* memiliki kelemahan keamanan.
2. Kode *PHP* bisa dibaca oleh semua orang.
3. Biaya untuk encoding membutuhkan biaya yang sangat mahal.

1.3 Metode Yang Digunakan

Teknik pengumpulan data merupakan langkah dalam menganalisis sistem, karena tujuan utama dari penelitian adalah mengumpulkan data. Teknik pengumpulan data bisa dilakukan dengan beberapa cara seperti :

1.3.1 Wawancara

Wawancara adalah suatu bentuk tanya-jawab dengan narasumber dengan tujuan mendapatkan keterangan, penjelasan, pendapat, fakta, bukti tentang suatu masalah atau suatu peristiwa, pengumpulan data dengan menggunakan wawancara mempunyai beberapa keuntungan sebagai berikut :

1. Lebih mudah dalam mengenali bagian *System* mana yang dianggap baik dan bagian mana yang dianggap kurang baik.
2. Jika ada bagian tertentu yang perlu digali lebih dalam, maka dapat langsung menanyakan kepada narasumber.
3. Dapat menggali kebutuhan *Users* secara lebih bebas.
4. *Users* dapat mengungkapkan kebutuhannya secara lebih bebas

Selain mempunyai beberapa kelebihan, Teknik wawancara juga mempunyai beberapa kelemahan. Berikut ini adalah beberapa kelemahan dari teknik wawancara sebagai berikut :

1. Wawancara akan sulit dilakukan jika narasumber kurang dapat mengungkapkan kebutuhannya.
2. Pertanyaan dapat menjadi tidak terarah, terlalu fokus pada hal-hal tertentu dan mengabaikan bagian lainnya.

1.3.2 Pengamatan/Observasi

Teknik pengamatan/observasi adalah suatu teknik pengumpulan data dengan cara menghimpun bahan-bahan keterangan yang dilakukan dengan mengadakan pengamatan dan pencatatan secara sistematis terhadap fenomena-fenomena yang dijadikan obyek pengamatan

Pengumpulan data dengan menggunakan pengamatan/observasi memiliki keuntungan sebagai berikut :

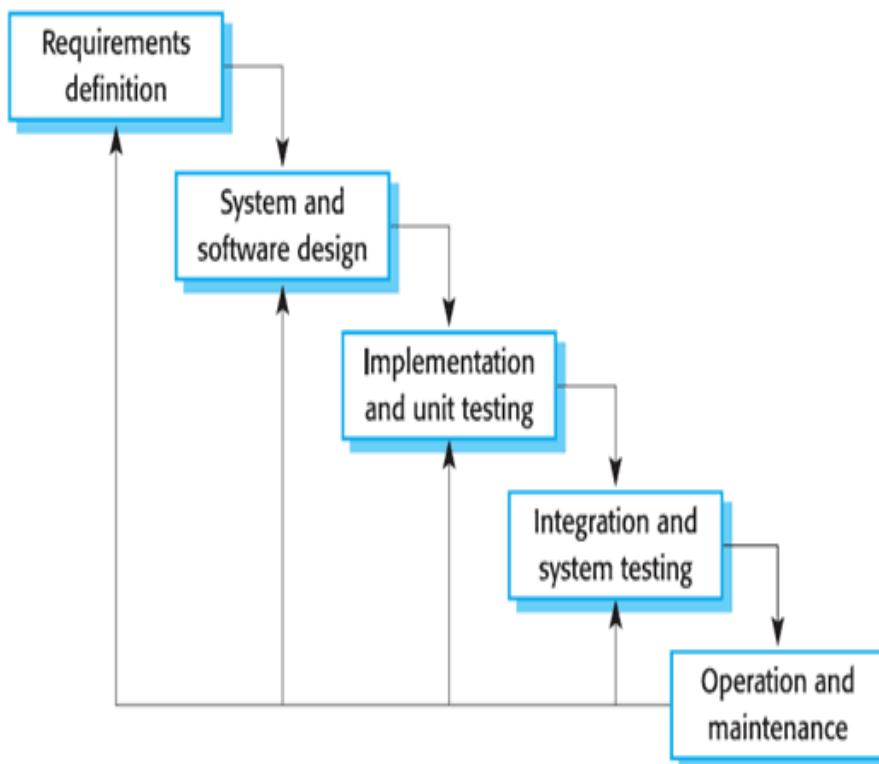
1. Analisis dapat melihat langsung bagaimana *System* lama berjalan.
2. Mampu menghasilkan gambar lebih baik jika dibanding dengan teknik lainnya.

Sedangkan kelemahan menggunakan teknik pengamatan/observasi adalah sebagai berikut:

1. Memerlukan waktu cukup lama karena jika observasi waktunya sangat terbatas maka gambaran *System* secara keseluruhan akan sulit untuk diperoleh
2. Orang-orang yang sedang diamati biasanya perilakunya akan berbeda dengan perilaku sehari-hari (cenderung berusaha terlihat baik). Hal ini akan menyebabkan gambaran yang diperoleh selama observasi akan berbeda dengan perilaku sehari-hari.

1.3.3 Waterfall Model

Definisi Metode *Waterfall Model* pengembangan software yang diperkenalkan oleh Winston Royce pada tahun 70-an ini merupakan model klasik yang sederhana dengan aliran sistem yang linier — keluaran dari tahap sebelumnya merupakan masukan untuk tahap berikutnya. Pengembangan dengan model ini adalah hasil adaptasi dari pengembangan perangkat keras, karena pada waktu itu belum terdapat metodologi pengembangan perangkat lunak yang lain. Proses pengembangan yang sangat terstruktur ini membuat potensi kerugian akibat kesalahan pada proses sebelumnya sangat besar dan acap kali mahal karena membengkaknya biaya pengembangan ulang



Gambar 1. 4 Metode *Waterfall*

Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak berurutan, di mana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian. Dalam pengembangannya metode *Waterfall* memiliki beberapa tahapan yang runtut: requirement (analisis kebutuhan), *design* sistem (*System design*), Coding & Testing, Penerapan Program, pemeliharaan.

1.3.4 Tahap-Tahap Metode *Waterfall Model*

Dalam pengembangan perangkat lunak menggunakan metode *Waterfall Model* memiliki 5 tahapan sebagai berikut

1. *Requirement (analisis kebutuhan)*

Dalam langkah ini merupakan analisis terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study* literatur. Seseorang *System* analisis akan menggali informasi sebanyak-banyaknya dari *Users* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *Users* tersebut. Tahapan ini akan menghasilkan dokumen *Users* requirement atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *Users* dalam pembuatan sistem. 14 | Jurnal Teknologi Informasi ESIT Vol. XII No. 01 April 2018 Dokumen inilah yang akan menjadi acuan *System* analisis untuk menerjemahkan kedalam bahasa pemrograman.

2. Design System (design sistem)

Proses *design* akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat koding. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut software requirement. Dokumen inilah yang akan digunakan programmer untuk melakukan aktivitas pembuatan sistemnya.

3. Coding & Testing (penulisan sinkode program / implemention)

Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh programmer yang akan menerjemahkan transaksi yang diminta oleh *Users*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan *computer* akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan testing terhadap sistem yang telah dibuat tadi. Tujuan testing adalah menemukan kesalahan-kesalahan terhadap *System* tersebut dan kemudian bisa diperbaiki.

4. Transition Phase

Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, *design* dan pengkodean maka sistem yang sudah jadikan digunakan oleh *Users*.

5. Pemeliharaan (*Operation & Maintenance*)

Perangkat lunak yang susah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (peripheral atau *System* operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

Keuntungan Metode *Waterfall* :

1. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.
2. Document pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya. Jadi setiap fase atau tahapan akan mempunyai dokumen tertentu.

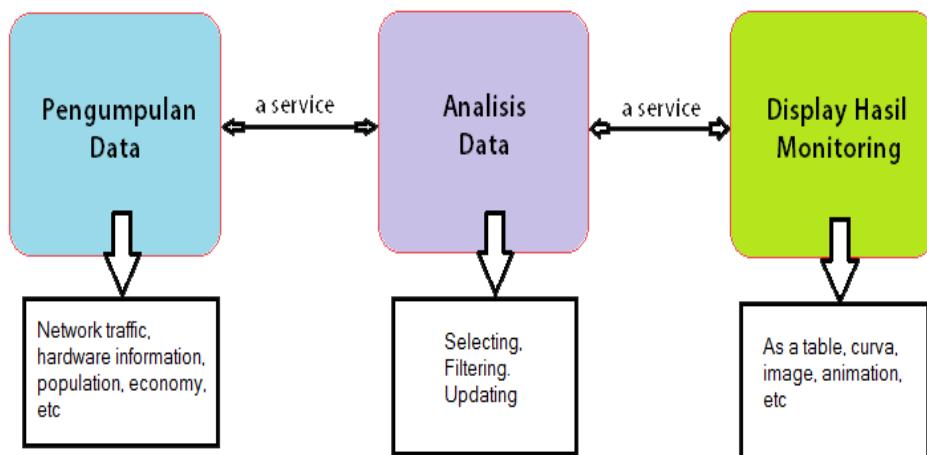
Kelemahan Metode *Waterfall* :

1. Diperlukan manajemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.
2. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan.

3. Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidakpastian pada saat awal pengembangan.

1.4 Sistem *Monitoring*

Sistem *Monitoring* merupakan suatu proses untuk mengumpulkan data dari berbagai sumber daya. Biasanya data yang dikumpulkan merupakan data yang real time. Secara garis besar tahapan dalam sebuah sistem *Monitoring* terbagi ke dalam tiga proses besar seperti yang terlihat pada gambar berikut :



Gambar 1. 5 Proses dalam sistem *Monitoring*

Proses-proses yang terjadi pada suatu sistem Monitoring dimulai dari pengumpulan data seperti data dari network traffic, hardware information, dan lain-lain yang kemudian data tersebut dianalisis pada proses analisis data dan pada akhirnya data tersebut akan ditampilkan.

1.5 Job Description

Job Description adalah rekaman tertulis mengenai tanggung jawab dari pekerjaan tertentu. Dokumen ini *Menunjukan* kualifikasi yang dibutuhkan untuk bagian lain dalam perusahaan. Sehingga dapat disimpulkan bahwa deskripsi pekerjaan adalah pernyataan tertulis mengenai gambaran suatu pekerjaan, kondisinya, dan hubungannya dengan bagian lain dalam organisasi.

1.5.1 Pengertian *Job Description*

Job Description adalah analisa jabatan yang merupakan suatu aktivitas dalam menentukan apa pekerjaan yang harus dilakukan dan siapa yang harus melakukan tugas tersebut. Aktivitas ini adalah sebuah upaya untuk menciptakan kualitas dari pekerjaan dan kualitas dari kinerja total dari sebuah perusahaan. Perusahaan akan baik jika sumber daya manusia didalamnya telah mampu melaksanakan pekerjaan masing-masing dengan jelas, spesifik, serta tidak memiliki peran ganda yang dapat menghambat proses pencapaian kinerja. Pekerjaan atau tugas dilakukan maka sebelumnya, terlebih dahulu harus dilakukan suatu *Job Analysis* (analisis pekerjaan) perlu diingat bahwa analisis pekerjaan tidak sama dengan *Job Description* namun pada akhirnya analisis pekerjaan akan berakhir pada suatu bentuk uraian tugas. Analisis pekerjaan merupakan kegiatan atau proses menghimpun dan menyusun berbagai informasi yang berkaitan dengan setiap pekerjaan, tugas-tugas, jenis pekerjaan, dan tanggung jawabnya secara operasional untuk mewujudkan tujuan organisasi atau bisnis sebuah perusahaan.

Menurut Tim Penulis TSM menyatakan:

“Job Description is a systematic summaries of information gained from notes taken and record in the job analysis process” (Uraian tugas yaitu uraian tentang semua yang dikerjakan oleh pemegang jabatan dalam menjalankan tugas-tugas jabatan)

Menurut Doni Juni Priansa menyatakan:

“ Deskripsi pekerjaan mendefinisikan apa yang harus diperlukan pimpinan untuk melaksanakan kegiatan, tugas, atau pekerjaannya ”.

Menurut Gary Desler menyatakan bahwa:

“Deskripsi pekerjaan adalah daftar jabatan, tanggung jawab, hubungan pelaporan, kondisi jabatan dan tanggung jawab “.

Berdasarkan beberapa pendapat para ahli di atas, nampak bahwa deskripsi pekerjaan merupakan keterangan singkat mengenai tugas dan tanggung jawab dari suatu jabatan. Deskripsi pekerjaan merupakan pedoman, petunjuk dan arah tindakan bagi tenaga kerja untuk melaksanakan pekerjaan sesuai dengan tugas dan tanggung jawabnya. Oleh karena itu, dengan adanya deskripsi pekerjaan diharapkan karyawan dapat melaksanakan tugas dengan baik guna menciptakan kinerja yang optimal.

1.5.2 Manfaat *Job Description*

Job Description adalah kumpulan informasi tertulis tentang suatu jabatan, tanggung jawab dan wewenang maka *Menurut Ardana, Mujiati dan Utama*, deskripsi pekerjaan tersebut dapat memberikan manfaat sebagai berikut:

Deskripsi pekerjaan membantu menghindari adanya kebingungan pekerjaan dan memberikan pemahaman dalam melaksanakan pekerjaan

1. Dapat menghindari tumpang tindih tanggung jawab dalam melaksanakan tugas.
2. Memudahkan prosedur rekrutmen, seleksi, pelatihan dan berbagai aktivitas SDM
3. Membantu pegawai dalam merencanakan karier, mengurangi praktik diskriminasi dalam promosi dan pemindahan serta memudahkan evaluasi dalam pekerjaan untuk memastikan adanya keadilan dalam pemberian kompensasi.
4. Bermanfaat dalam program keselamatan kerja dapat *Menunjukkan* tindakan berbahaya dan mengadakan perubahan dalam pelaksanaan.
5. Deskripsi pekerjaan penting dalam perencanaan perubahan organisasi dan reorganisasi sesuai perkembangan keadaan.
6. Memberi arahan tentang pengalaman yang diperlukan untuk melaksanakan pekerjaan.

1.5.3 Prinsip-Prinsip *Job Description*

Job Description ataupun uraian pekerjaan merupakan dokumen penting untuk memandu proses seleksi yang mana digunakan untuk memberikan pekerjaan tersebut kepada calon pegawai yang berpotensi. Oleh karena itu, uraian pekerjaan hendaknya dapat mendeskripsikan dengan tepat isi pekerjaan, lingkungan, dan kondisi pekerjaan. *Menurut Ardana, Mujiati dan Utama, terdapat enam kualifikasi yang harus diperhatikan dalam pembuatan uraian jabatan, antara lain :*

1. Sistematis, artinya deskripsi pekerjaan terdiri dari komponen-komponen yang mempunyai fungsi dan tersusun dalam tata hubungan yang membentuk suatu sistem sehingga mudah dipahami.
2. Jelas, artinya deskripsi pekerjaan dapat memberikan pembacanya isi dan maksud yang jelas, terang, gemilang, dan tidak meragukan. Sehingga mudah dipahami dan diterapkan oleh setiap pemangku jabatan di dalam perusahaan.
3. Ringkas, artinya deskripsi pekerjaan perlu menggunakan kata-kata dan kalimat yang singkat, pendek sehingga pembaca tidak memerlukan waktu yang lama untuk membaca dan memahaminya.
4. Tepat, artinya deskripsi pekerjaan dapat menyajikan uraian yang cocok, sesuai dan tepat seperti apa yang dimaksud oleh isi pekerjaan sehingga pembaca dapat memperoleh gambaran yang sama dengan isi yang sebenarnya.

5. Taat azas, artinya deskripsi pekerjaan berisi kata dan kalimat yang isinya *Menunjukkan arah dan maksud yang sama atau selaras dan tidak bertentangan satu sama lain.*
6. Akurat, artinya deskripsi pekerjaan disusun secara teliti, dengan memaparkan keadaan yang lengkap, tidak kurang dan tidak lebih

1.5.4 Penyusunan *Job Description*

Sebelum *Job Description* diterapkan ada beberapa hal yang harus dilakukan sebagai berikut ini :

A. *Job Analys* (Analisis Pekerjaan)

Untuk mendapatkan karyawan yang berkualitas dan berkuantitas baik, sehingga efektif mengerjakan tugas-tugasnya harus dilakukan dengan cara analisis jabatan (*job analysis*), uraian jabatan (*Job Description*) dan spesifikasi pekerjaan (*job specification*). Dengan analisis jabatan, uraian pekerjaan dan spesifikasi pekerjaan maka dapatlah ditentukan kualitas dan kuantitas pegawai yang dibutuhkan.

Kehadiran struktur organisasi mutlak ada di dalam suatu kegiatan bisnis dan menjadi komponen penting dari manajemen organisasi. Kadangkala struktur organisasi yang disusun hanya sekedar pajangan saja sehingga implementasinya dapat mengacaukan pekerjaan di antara sesama pekerja. Penyusunan struktur organisasi harus sesuai dengan kebutuhan. Struktur organisasi yang gemuk bila tidak berfungsi menjadi sia-sia, demikian pula struktur organisasi yang kurus bila itu dalam struktur organisasi perlu menempatkan orang-orangnya yang mempunyai keterampilan (kompetensi) sesuai dengan bidang pekerjaannya dan yang tak kalah penting terdapat pembagian pekerjaan (*Job Description*) yang jelas dan tegas sebagai pedoman kerja. Didalam struktur organisasi, orang-orangnya harus mempunyai tugas,

wewenang dan tanggung jawab didalam departemen (unit kerjanya) masing-masing, terdapat tata hubungan diantara departemen, koordinasi internal dan eksternal, dan fungsi kontrol dari manajemen. Esensi dalam struktur organisasi adalah adanya pedoman kerja. Struktur organisasi yang efektif akan tercipta bila pembagian tugas berjalan dengan baik sebagai pedoman kerja yang bermuara pada peningkatan produktivitas perusahaan. Berikut ini akan 30 dijelaskan mengenai pengertian *Job Analys* (Analisis Pekerjaan) *Menurut* para ahli yaitu :

Menurut Gomes Menyatakan bahwa :

“Analisis jabatan adalah proses pengumpulan informasi mengenai suatu pekerjaan yang dilakukan seorang pekerja, yang dilaksanakan dengan mengamati atau mengadakan interView pada pekerjaan, dengan bukti-bukti yang benar dari supervisor”.

Bernardin & Russell menyatakan bahwa :

“Analisis pekerjaan ini akan menghasilkan daftar uraian pekerjaan pernyataan tertulis mengenai kewajiban-kewajiban pekerja dan bisa juga mencakup standar kualifikasi, yang merinci pendidikan dan pengalaman minimal yang diperlukan bagi seorang pekerja untuk melaksana kan kewajiban dari kedudukannya secara memuaskan”.

Berdasarkan pendapat *Menurut* para ahli diatas dapat disimpulkan bahwa analisis jabatan (*Job Analysis*) adalah suatu bentuk pengembangan uraian terperinci dari tugas-tugas yang harus dilakukan dalam suatu jabatan, penentuan hubungan dari satu jabatan dengan jabatan lain yang ada, dan penentuan tentang pengetahuan, ketampilan, dan kemampuan-kemampuan lain yang diperlukan karyawan untuk melakukan pekerjaan secara efisien dan

efektif sebelum dibentuknya suatu struktur organisasi dan *Job Description* (uraian tugas).

Manfaat Analisis Pekerjaan (*Job Analysis*)

Manfaat dari analisis pekerjaan ini mengacu pada Visi,Misi, Tujuan dan Strategi - Strategi perusahaan/ organisasi, maka analisis pekerjaan merupakan hal-hal yang menggambarkan apa saja yang harus dilakukan perusahaan untuk mencapai tujuannya, sehingga bermanfaat. Dan mengupayakan terjadinya ketidaksesuaian pekerjaan yang dilakukan dengan penetapan tujuan yang telah ditetapkan sebelumnya. Secara lebih rinci Bilson Simamora menjelaskan mengenai manfaat dari analisis pekerjaan yaitu:

1. Analisis penyusunan kepegawaian

Dalam analisis penyusunan kepegawaian, seorang manajer akan mencari informasi tentang pekerjaan apa saja yang harus dilakukan dan mengolahnya guna menyusun struktur kepegawaian, sehingga pelayagunaan para karyawan akan lebih optimal serta mencapai tujuan yang lebih efektif dan efisien.

2. Desain organisasi

Analisis pekerjaan sering diaplikasikan dalam desain dan redesain pekerjaan tertentu serta pekerjaan terkait lainnya yang tujuannya adalah untuk menciptakan perubahan perilaku organisasional yang signifikan.

3. Redesain pekerjaan

Suatu pekerjaan dapat dikaji atau dikaji ulang ketika pekerjaan itu telah dipakai untuk meningkatkan metode pekerjaan, mengurangi kesalahan, mengeliminasi penanganan bahan yang tidak perlu dan duplikasi upaya mengurangi kelelahan.

4. Perencanaan kinerja

analisis pekerjaan menciptakan informasi, perencanaan dan evaluasi kinerja lebih meningkatkan tanggung jawab dan akhirnya memperbaiki kinerja karyawan, akurat dan hal itu merupakan hal yang bersifat fundamental. Kesuksesan manajemen tidak lepas dari analisis pekerjaan/jabatan karena profil pekerjaan yang dibuat akan menentukan apa saja aktifitas utama, penempatan kerja dengan persyaratan - persyaratan apa saja yang dibutuhkan dan pertanggung jawaban dari setiap pekerja/pegawai.

5. Pelatihan dan pengembangan

Lewat analisis pekerjaan, progam pelatihan dan pengembangan dari setiap lini dan lapisan organisasi akan mudah ditentukan. Deskripsi tugas pekerjaan merupakan materi yang membantu pembuatan isi program pelatihan. Sehingga perusahaan dapat mengetahui apa saja kebutuhan-kebutuhan yang diperlukan.

6. Jalur karir

Jalur karir merupakan penjabaran secara eksplisit dari urusan alternatif pekerjaan yang dapat diduduki oleh seorang individu dalam suatu karir organisasional. Ketika karyawan memiliki informasi tentang

persyaratan pekerjaan, maka akan memungkinkan karyawan akan memiliki perencanaan karir yang baik.

7. Kriteria seleksi

Output dari analisa pekerjaan akan menjadi dasar kriteria seleksi karyawan baik pada awal rekrutmen maupun ketika akan dipromOSikan atau penugasan selanjutnya.

8. Evaluasi pekerjaan

Analisis pekerjaan memberikan deskripsi atau gambarab tentang suatu pekerjaan bagaimana kekuatan dan kelemahan dari suatu organisasi.

Analisis jabatan akan dimulai dengan proses pengumpulan data-data dari intern atau langsung pada penilaian terhadap sistem yang ada. Sehingga dapat mengetahui data-data yang biasanya digunakan adalah dokumen visi misi perusahaan, dokumen peraturan perusahaan atau perjanjian kerja bersama, dokumen prosedur operasi yang sudah distandardisasikan, dan dokumen lain yang dapat memberikan histori jabatan yang akan dianalisa. Selanjutnya analis jabatan (sebutan untuk orang yang melakukan analisa jabatan) akan mempelajari dokumen tersebut dan menggali informasi mengenai suatu jabatan. Setelah mendapatkan info sebagai berikut :

1. Penyebaran kusioner
2. Wawancara dengan pemangku jabatan, atasan,rekan,dan bawahan.
3. Observasi. Proses terjun di lapangan ini dilakukan untuk organisasi yang proses bisnisnya sudah berjalan.

Artinya sudah ada kegiatan yang dilakukan. Di lapangan ini, analis jabatan akan menggali mengena tanggungjawab dan wewenang jabatan, tugas-tugas yang dijalankan, prosedur standar dalam operasionalisasi, kendala dan hambatan, pihak yang biasa terlibat dalam penanganan pekerjaan.

B. *Job Description* (Uraian Pekerjaan)

Deskripsi pekerjaan merupakan langkah pertama dari proses analisa jabatan, yaitu gambaran jabatan yang tersedia, pernyataan yang akurat dan ringkas mengenai apa saja yang diharapkan oleh karyawam didalam pekerjaan, menggambarkan tugas- tugas yang dilaksanakan oleh pemangku jabatan. Dengan adanya deskripsi jabatan diharapkan setiap karyawan yang memegang jabatan tertentu akan mengetahui terjadinya penyimpangan dan pekerjaan ganda antara jabatan satu dengan yang lainnya. Oleh karena itu sebaiknya deskripsi pekerjaan dibuat sedemikian rupa agar mudah dipahami dan dimengerti oleh pegawai.

C. *Job Specification* (Spesifikasi Pekerjaan)

Spesifikasi pekerjaan (*job specification*) disusun berdasarkan uraian pekerjaan dengan menjawab pertanyaan tentang ciri, karakteristik, pendidikan, pengalaman dan yang lainnya dari orang yang akan melaksanakan pekerjaan tersebut dengan baik. Spesifikasi pekerjaan *Menunjukan persyaratan orang yg akan direkrut.*

Menurut Hasibuan (2010:34) menyatakan bahwa :

“Spesifikasi jabatan adalah uraian persyaratan kualitas minimum orang yang bisa diterima agar dapat menjalankan satu jabatan dengan baik dan kompeten, juga memuat ringkasan yang jelas dan kualitas definitif yang dibutuhkan dari pemangku jabatan itu”.

Menurut Henry Simamora (2010) menyatakan bahwa :

” Spesifikasi pekerjaan adalah uraian persyaratan kualitas minimum orang yang bisa diterima agar dapat menjalankan satu jabatan dengan baik dan kompeten”.

Pada umumnya spesifikasi pekerjaan memuat ringkasan pekerjaan yang jelas dan kualitas definitif yang dibutuhkan dari pemangku jabatan itu. Spesifikasi pekerjaan memberikan uraian informasi mengenai hal-hal berikut:

1. Tingkat pendidikan pekerja.
2. Jenis kelamin pekerja.
3. Keadaan fisik pekerja.
4. Pengetahuan dan kecakapan pekerja.
5. Batas umur pekerja.
6. Minat pekerja.
7. EmOSI dan temperamen pekerja.
8. Pengalaman pekerja.

Perbedaan Job analys, *Job Description* dengan Job Specification :

- a. Analisis pekerjaan merupakan proses pengumpulan dan pemeriksaan atas aktifitas kerja. Deskripsi pekerjaan merupakan dokumen yang menyediakan informasi mengenai kewajiban, tugas, dan tanggung jawab dari pekerjaan. Spesifikasi pekerjaan merupakan keahlian, pengetahuan, dan kemampuan yang dibutuhkan untuk melaksanakan pekerjaan.

- b. Deskripsi pekerjaan lebih berhubungan dengan organisasi, struktur, tanggung jawab, dan hubungan diantaranya. Deskripsi pekerjaan merupakan peta organisasional yang *Menunjukkan tujuan pekerjaan dan apa yang harus dikerjakan untuk mencapai tujuan organisasi.*
- c. Spesifikasi pekerjaan lebih menekankan pada persyaratan fisik, pengetahuan, pengalaman, pendidikan, kemampuan gerak, dan fisiologis, dan kecerdasan yang diperlukan untuk melaksanakan tanggung jawab yang dibebankan pada pekerjaan

Kaitan antara *job analysis*, *Job Description* dan *job specification* adalah *job analysis* merupakan proses mencari informasi sampai dengan analisis untuk penyusunan uraian pekerjaan (*Job Description*) dan spesifikasi pekerjaan atau persyaratan apa/siapa dari masing-masing jenis pekerjaan harus dikerjakan. Uraian yang diperoleh dari analisis pekerjaan diharapkan benar-benar diperoleh sesuai dengan kebutuhan organisasi atau lembaga sehingga efektifitas dan efesiensi SDM

1.5.5 Dimensi dan Indikator *Job Description*

Job Description adalah tugas apa saja yang harus dilakukan oleh setiap karyawannya didalam perusahaan. Adapun *Menurut Robbins dan Judge* oleh Diana Angelica *Job Description* yaitu skema tertulis mengenai terjemahan tanggung jawab dari pekerjaan tertentu. Sehingga dapat disimpulkan bahwa deskripsi pekerjaan adalah penyataan tertulis mengenai gambaran suatu pekerjaan, kondisi dan hubungannya dgn bagian lain. Dimensi dan indikator *Job Description* adalah :

Tabel 1. 1 Dimensi dan Indikator *Job Description*

No	Dimensi	Indikator
1.	Wewenang	<ul style="list-style-type: none"> 1. Kewenangan teridentifikasi secara jelas 2. Tidak overlapping dengan pOSIsi lain 3. Kesesuaian wewenang dengan pOSIsi
2.	Tanggung Jawab	<ul style="list-style-type: none"> 1. Memperoleh kejelasan mengenai tanggung jawab yang diemban secara keseluruhan 2. Arah pertanggung jawaban jelas 3. Kompetensi yan diberikan sesuai dengan tanggung jawab pekerjaan
3.	Kondisi Pekerjaan	<ul style="list-style-type: none"> 1. Peraturan atau kebijakan perusahaan dapat dipahami 2. Adanya kejelasan koodinasi dalam melaksanakan pekerjaan
4.	Fasilitas Kerja	<ul style="list-style-type: none"> 1. Kelengkapan fasilitas untuk mendukung kelancaran pekerjaan 2. Kesesuaian fasilitas dengan kebutuhan pekerjaan
5.	Standar Hasil Kerja	<ul style="list-style-type: none"> 1. Kejelasan mengenai target yang diharapkan 2. Kesesuaian target dengan bidang pekerjaan
6.	Pendidikan	<ul style="list-style-type: none"> 1. Kesesuaian tanggung jawab pekerjaan denga latar belakang pendidikan 2. Kesesuaian tanggung jawab pekerjaan dengan latar belakang pengalaman kerja

7. Kompetensi
1. Kesesuaian pekerjaan dengan pengetahuan
 2. Kesesuaian pekerjaan dengan keahlian dan keterampilan

1.5.6 Kerangka Pemikiran

Sumber daya manusia menjadi dan penentu jalannya suatu organisasi, maka perhatian manajemen pun mutlak diperlukan agar aktivitas organisasi berjalan pada koridor atau tujuan yang telah ditetapkan. Perhatian terhadap aspek-aspek yang akan *Menunjang* kesuksesan organisasi penting dilakukan, karena kemajuan organisasi ditentukan oleh kinerja dan keefektifan para karyawan dalam menjalankan tugas pokonya. Setiap organisasi mengharapkan karyawannya mampu melaksanakan tugasnya secara efektif dan efesien. Namun semua itu tidak akan kepuasan kerja karyawan menjadi sangat penting karena kepuasan kerja adalah hal yang menjadi penyebab dan pendukung perilaku manusia, agar bersedia menyalurkan tenaga dan pikirannya untuk bekerja dengan sungguh-sungguh dan antusias mencapai hasil yang optimal sebagai bentuk tanggung jawabnya. Tanpa adanya kepuasan kerja dalam dirinya, maka dapat dipastikan bahwa karyawan tidak akan mampu menghasilkan hasil yang maksimal. Dalam dunia organisasi, semua tujuan tidak akan tercapai apabila dalam pelaksanaannya dilakukan oleh karyawan yang kurang atau tidak memiliki kepuasan kerja yang tinggi.

1.5.7 OHC (*Operational Human Capital*)

Human capital merupakan segenap pengetahuan, keahlian, keterampilan, dan kreativitas yang diwujudkan dalam kemampuan kerja yang dapat digunakan untuk menghasilkan layanan profesional dan nilai ekonomi. Human capital pada prinsipnya menjadi bagian dari manajemen sumber daya manusia, hanya saja pengelolaan dan pengembangan kemampuan manusia sebagai sumber daya lebih berfokus pada peningkatan pengetahuan dan keterampilan yang dapat mendukung pengembangan organisasi atau perusahaan. *Human capital* menempatkan sumber daya manusia pada level derajat yang lebih tinggi dari sekadar sumber daya, tetapi aset berharga yang bernilai dan bermanfaat bagi organisasi atau perusahaan. Sebagai aset berharga, sumber daya manusia bahkan dapat dibandingkan dengan portfolio investasi yang dapat dikembangkan dan dilipatgandakan kinerjanya. Sebab itu, *human capital* tidak memandang sumber daya manusia sebagai liabilitas atau biaya yang membebani keuangan dan mengurangi tingkat keuntungan organisasi. Muncul dan berkembangnya konsep *human capital* dipicu oleh pemanfaatan karyawan sebagai sumber daya sebagai maksimal. Artinya, karyawan harus siap untuk ‘diperas’ pengetahuan, keahlian, keterampilan, dan kreativitasnya untuk kepentingan perkembangan dan kemajuan organisasi. Sebab, sebagai sumber daya, karyawan akan mengalami penurunan produktivitas seiring dengan bertambahnya usia. Bahkan, tak sedikit karyawan yang merasa telah berada pada zona nyaman justru kinerjanya mengendur. Sebelum sumber daya manusia ini ‘habis’, organisasi atau perusahaan berusaha menambah dan menciptakan *value* untuk mengembangkan sumber daya manusianya. Perusahaan mengelola karyawan sedemikian rupa guna meningkatkan kemampuan kerja sehingga dapat mencapai kinerja pada tingkat yang lebih tinggi. Pengelolaan ini bisa dilakukan dengan menciptakan modal intelektual serta kecerdasan komunikasi dan interaksi dengan karyawan

lain melalui pendidikan dan pelatihan secara intensif sehingga dapat menghasilkan pengetahuan yang bermanfaat bagi pengembangan perusahaan. Seiring dengan berkembangnya pengetahuan dan kinerja karyawan, perusahaan juga tidak abai dalam memberikan *reward* guna meningkatkan loyalitas karyawan. Pemberian *reward* dan *benefit* menarik seperti tunjangan, bonus tahunan, dan fasilitas lain tak hanya akan memotivasi karyawan saja, tetapi juga memicu kesadaran mereka untuk tetap bertahan di perusahaan.

1.5.7.1 Jenis-jenis *Human Capital*

Seperti halnya dengan sumber daya keuangan dan peralatan, konsep *human capital* menempatkan sumber daya manusia sebagai aset berharga yang tak kalah penting. Namun, *human capital* sebagai aset tidaklah berwujud karena berupa pengetahuan, keahlian, keterampilan, dan kreativitas yang melekat pada sumber daya manusia. Keberadaan sumber daya manusia dengan segenap sifat kolektif yang melekat padanya memberi modal bagi perusahaan untuk meningkatkan produktivitasnya, menghasilkan layanan profesional, dan tentunya nilai ekonomi yang menguntungkan. Berkenaan dengan hal tersebut, *human capital* yang ada dan bisa dikembangkan dalam diri sumber daya manusia terdiri dari beberapa jenis sebagai berikut :

1. General management *human capital*

Jenis *human capital* ini dikembangkan untuk sumber daya manusia di level manajemen tinggi atau eksekutif yang meliputi kompetensi manajerial, kepemimpinan, kemampuan mengambil keputusan, dan keahlian fungsional, termasuk dalam mengumpulkan, mengolah, dan mengelola sumber daya keuangan, teknis, dan sumber daya manusia. Jenis *general management human capital* penting dimiliki dan dikembangkan sehingga para eksekutif perusahaan

mampu berinteraksi dengan investor dan kolega bisnis dalam memajukan perusahaan.

2. *Strategic human capital*

Strategic human capital merupakan jenis modal manusia yang mencakup keahlian dan keterampilan strategis yang didapatkan dari pengalaman menghadapi situasi tertentu. Misalnya kemampuan strategis dalam mengatur keuangan terkait pemotongan anggaran biaya untuk efisiensi sebab adanya situasi keuangan yang kurang menguntungkan. Selain itu, kemampuan bermanuver menghadapi situasi pasar yang tidak bisa diprediksi juga termasuk dalam jenis *human capital* ini. Intinya, *strategic human capital* dibutuhkan untuk beradaptasi dengan perubahan dan situasi di lingkungan baru.

3. *Industry human capital*

Segenap pengetahuan yang berkaitan dengan industri baik teknis, regulasi maupun supplier termasuk dalam jenis *industry human capital*. Namun, jenis *human capital* ini hanya berlaku untuk perusahaan yang beroperasi di bawah peraturan yang sama atau sejenis saja. Misalnya seperti industri otomotif, obat dan makanan, serta yang lainnya.

4. *Relationship human capital*

Kecerdasan berkomunikasi *Menunjukkan* kemampuan menjalin komunikasi dan berinteraksi dengan orang lain baik di dalam maupun di luar tim kerja. Kemampuan jenis ini termasuk dalam *relationship human capital*. Meski tampak sederhana, namun kemampuan ini harus dimiliki oleh setiap karyawan sehingga

perusahaan harus memanfaatkan dan mengembangkan potensinya secara maksimal. Karyawan yang memiliki kemampuan yang baik dalam berkomunikasi dan berinteraksi dapat mendukung kelancaran pelaksanaan tugas sehingga tercapainya kinerja yang tinggi dapat diraih dengan lebih mudah.

5. *Company specific human capital*

Setiap perusahaan pasti memiliki struktur dan budaya kerja yang berbeda, termasuk juga peraturan dan kebijakannya. Setiap karyawan dituntut untuk mampu beradaptasi dengan segala sesuatu yang menjadi ketentuan internal perusahaan tersebut. Inilah yang disebut dengan *company specific human capital*. Karyawan yang mengenal dengan baik segala atribut yang melekat pada perusahaan, tentu akan lebih mudah menjalankan tugas sesuai dengan visi dan misi yang mengarah pada kemajuan perusahaan.

Pada prinsipnya, *human capital* menjadi bagian dari manajemen sumber daya manusia. Hanya saja, *human capital* memandang dan memperlakukan karyawan bukan hanya sekadar sumber daya manusia yang mendukung pelaksanaan tugas atau operasional perusahaan, tetapi juga sekaligus menjadi aset atau modal utama bagi perusahaan dalam mencapai tujuannya. Sebagai modal, *human capital* cenderung tak berwujud karena berupa pengetahuan, keahlian, keterampilan, dan kecerdasan yang melekat pada sumber daya manusia. Meski demikian, peranannya tak kalah penting dengan sumber daya keuangan dan peralatan yang dimiliki perusahaan.

BAB II

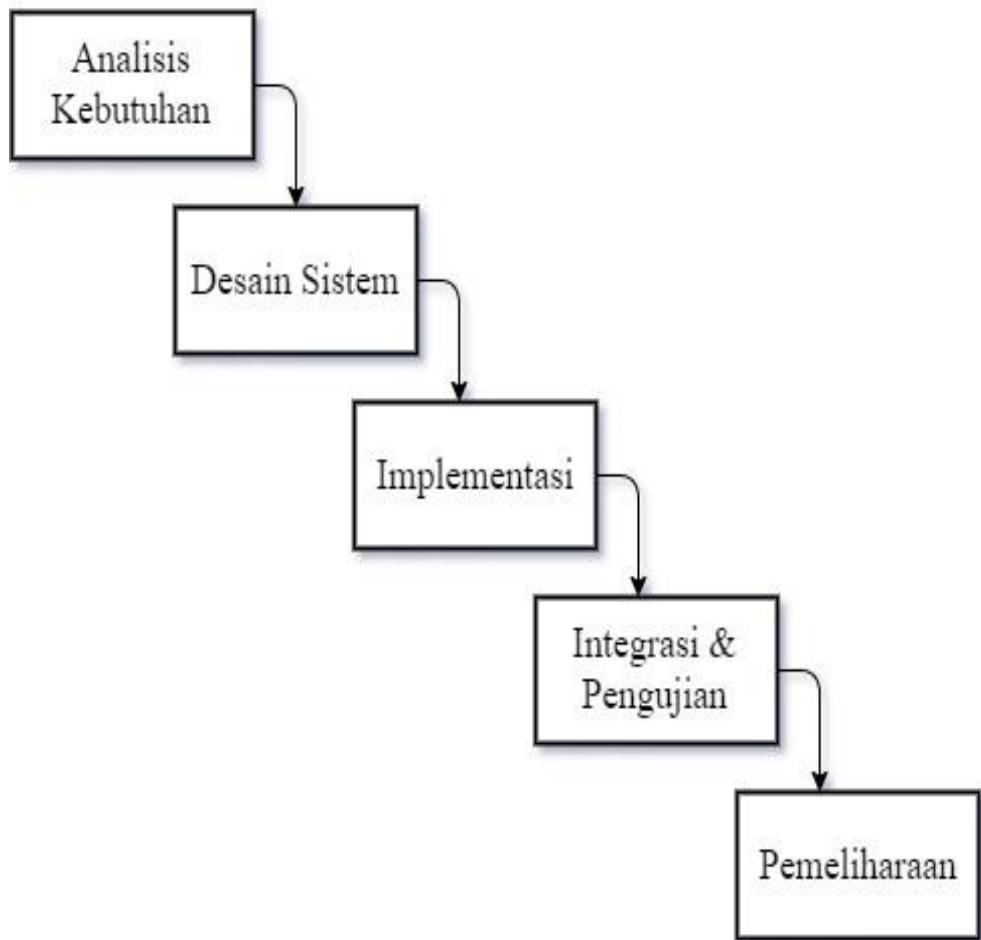
METODELOGI PENELITIAN

Metodologi penelitian adalah sebuah proses dari prinsip dan prosedur yang digunakan untuk mendekati masalah dalam mencari suatu solusi atau jawaban. Dengan kata lain, metodologi adalah suatu pendekatan umum yang digunakan untuk mengkaji sesuatu topik penelitian. Hal-hal yang akan memengaruhi hasil dari penelitian yaitu salah satunya adalah keakuratan hasil dari penelitian tersebut. Hal ini sangat dipengaruhi oleh metode analisa dan pengolahan yang digunakan dalam pembuatan sebuah penelitian.

Dalam analisis permasalahan dilakukan observasi terhadap sistem yang sedang berjalan dengan tujuan untuk menggali informasi, dan mengetahui segala permasalahan yang ada pada sistem. Permasalahan utama pada sistem manual adalah pimpinan perusahaan kesulitan untuk melakukan pengawasan terhadap semua anak cabangnya. Terlebih apabila perusahaan tersebut memiliki banyak anak cabang dan lokasinya saling berjauhan. Pimpinan perusahaan membutuhkan informasi yang cepat dan akurat untuk dapat membuat keputusan strategis dan taktis bagi kemajuan perusahaan. Informasi yang lengkap dan cepat tentang aktivitas operasional dan progress pekerjaan di masing-masing anak perusahaan tersebut menjadi hal yang mustahil diperoleh oleh pemimpin perusahaan apabila tidak ada sistem informasi online yang mendukung kegiatannya.

2.1 DIAGRAM ALUR PENELITIAN

Berikut ini adalah Tahapan Metode *Waterfall Model*



Gambar 2. 1 *Diagram Metodelogi Penelitian*

Berdasarkan Gambar tersebut maka terdapat tahapan-tahapan dalam *diagram* alur metodologi penelitian akan dijelaskan sebagai berikut :

1. Analisis Kebutuhan

Seluruh kebutuhan software didapatkan pada fase ini, termasuk didalamnya kegunaan software yang diharapkan pengguna dan batasan software. Informasi ini biasanya dapat diperoleh melalui wawancara, survey atau diskusi. Informasi tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya.

2. Desain Sistem

Tahap ini dilakukan sebelum melakukan tahapan *coding system*. Tahap ini bertujuan untuk memberikan gambaran seperti apa sistem yang akan dibuat dan bagaimana interface untuk setiap kegiatannya. Tahap ini membantu dalam menspesifikasikan kebutuhan hardware dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan.

3. Implementasi

Pada tahap ini dilakukan tahapan pemrograman. Pembuatan software dipecah menjadi modul-modul kecil yang nantinya akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaan terhadap modul yang dibuat, apakah telah memenuhi fungsi yang diinginkan atau belum.

4. Interfrasi dan Pengujian

Penggabungan modul-modul yang telah dibuat dan dilakukan pengujian untuk mengetahui apakah software telah sesuai atau belum sesuai dengan desainnya.

5. Pemeliharaan

tahap terakhir dalam model *Waterfall*. Software yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya.

BAB III

PENJELASAN TOOLS DAN BAHASA PEMROGRAMAN YANG DIGUNAKAN

3.1 TOOLS YANG DIGUNAKAN

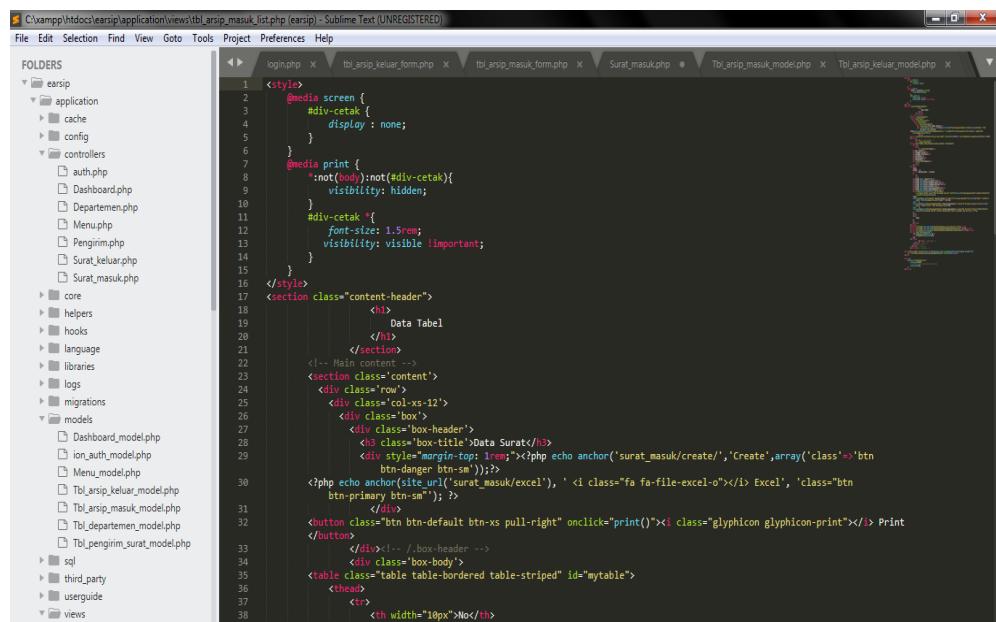
3.1.1 *Sublime*



Gambar 3. 1 Logo *Sublime*

Sublime Text adalah aplikasi editor untuk kode dan teks yang dapat berjalan diberbagai platform operating *System* dengan menggunakan teknologi Phyton API. Terciptanya aplikasi ini terinspirasi dari aplikasi Vim. Aplikasi ini sangatlah fleksibel dan powerfull. Fungsionalitas dari aplikasi ini dapat dikembangkan dengan menggunakan *Sublime-packages*. *Sublime Text* bukanlah aplikasi opensource dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, akan tetapi beberapa fitur

pengembangan fungsionalitas (packages) dari aplikasi ini merupakan hasil dari temuan dan mendapat dukungan penuh dari komunitas serta memiliki lisensi aplikasi gratis. *Sublime Text* mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur syntax highlight hamper di semua bahasa pemrogramman yang didukung ataupun dikembangkan oleh komunitas seperti; C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile and XML. Biasanya bagi bahasa pemrograman yang didukung ataupun belum terdukung secara *default* dapat lebih dimaksimalkan atau didukung dengan menggunakan *add-ons* yang bisa di *download* sesuai kebutuhan *Users*.



```

<style>
    @media screen {
        #div-cetak {
            display : none;
        }
    }
    @media print {
        *:not(body):not(#div-cetak){
            visibility: hidden;
        }
        #div-cetak {
            font-size: 1.5rem;
            visibility: visible !important;
        }
    }
</style>
<section class="content-header">
    <h1> Data Tabel </h1>
</section>
<!-- Main content -->
<section class="content">
    <div class="row">
        <div class="col-xs-12">
            <div class="box-header">
                <div class="box-title">Data Surat</h3>
                <div style="margin-top: 10px"><php echo anchor('surat_masuk/create','Create',array('class'))><br>
                <button type="button" class="btn btn-primary btn-sm">Excel</button>
                <?php echo anchor(site_url('surat_masuk/excel'), ' <i class="fa fa-file-excel-o"></i> Excel', ' class="btn btn-default btn-xs pull-right" onclick="print()"><i class="glyphicon glyphicon-print"></i> Print</button>
            </div>
            <div class="box-body">
                <table class="table table-bordered table-striped" id="mytable">
                    <thead>
                        <tr>
                            <th width="10px">No</th>
                            <th>Surat</th>
                        </tr>
                    </thead>
                    <tbody>
                </tbody>
            </table>
        </div>
    </div>
</div>

```

Gambar 3. 2 Contoh Tampilan Sublime Text

Berikut beberapa fitur yang diunggulkan dari aplikasi *Sublime Text*:

1. *Go to Anything*

Fitur yang sangat membantu dalam membuka *File* ataupun menjelajahi isi dari *File* hanya dengan beberapa *keystrokes*.

2. *Multiple Selections*

Fitur ini memungkinkan *Users* untuk mengubah secara interaktif banyak baris sekaligus, mengubah nama *variabel* dengan mudah, dan memanipulasi *File* lebih cepat dari sebelumnya.

3. *Command Pallete*

Dengan hanya beberapa *keystrokes*, *Users* dapat dengan cepat mencari fungsi yang diinginkan, tanpa harus me-navigasi melalui *Menu*.

4. *Distraction Free Mode*

Bila *Users* memerlukan fokus penuh pada aplikasi ini, fitur ini dapat membantu *Users* dengan memberikan tampilan layar penuh.

5. *Split Editing*

Dapatkan hasil yang maksimal dari *monitor* layar lebar dengan dukungan *editing* perpecahan. Meng>Edit sisi *File* dengan sisi, atau meng-Edit dua lokasi di satu *File*. Anda dapat meng-Edit dengan banyak baris dan kolom yang *Users* inginkan.

6. Instant Project Switch

Menangkap semua *File* yang dimasukkan kedalam *project* pada aplikasi ini. Terintegrasi dengan fitur *Goto Anything* untuk menjelajahi semua *File* yang ada ataupun untuk beralih ke *File* dalam *project* lainnya dengan cepat.

7. Plugin API

Dilengkapi dengan *plugin API* berbasis *Phyton* sehingga membuat aplikasi ini sangat Tangguh

8. Customize Anything

Aplikasi ini memberikan *Users* fleksibilitas dalam hal pengaturan fungsional dalam aplkasi ini

3.1.2 XAMPP



Gambar 3. 3 Logo XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak *System operasi*, merupakan kompilasi dari beberapa program. *XAMPP* merupakan *tool* 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 untuk anda atau auto konfigurasi. *XAMPP* merupakan salah satu paket *installasi Apache, PHP* dan *MySQL instant* yang dapat kita gunakan untuk membantu proses instalasi ketiga produk tersebut. Selain paket instalasi *instant XAMPP* versi 1.6.4 juga memberikan fasilitas pilihan pengunaan *PHP4* atau *PHP5*. Untuk berpindah versi *PHP* yang ingin digunakan juga sangat mudah dilakukan dengan menggunakan bantuan *PHP-Switch* yang telah disertakan oleh *XAMPP*, dan yang terpenting *XAMPP* bersifat *free* atau gratis untuk digunakan. *XAMPP* merupakan *tool* yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan meng-*install XAMPP* maka tidak perlu lagi melakukan instalasi dan konfigurasi *web Server Apache, PHP* dan *MySQL* secara manual. *XAMPP* akan menginstalasi dan mengkonfigurasikannya secara otomatis. Merupakan *web Server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis. Untuk mendapatkanya dapat men-*download* langsung dari *web* resminya. Fungsi *XAMPP* sendiri adalah sebagai *Server* yang berdiri sendiri (*localhost*), yang terdiri beberapa program antara lain: *Apache HTTP Server, MySQL Database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *XAMPP* sendiri merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL, PHP* dan *Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan *web Server* yang mudah untuk digunakan yang dapat menampilkan halaman *web* yang dinamis. Untuk mendapatkanya *XAMPP* anda dapat men-*download* langsung dari *web* resminya. Dan berikut beberapa definisi program lainnya yang terdapat dalam *XAMPP*.

Fitur-fitur *XAMPP*:

1. *Apache*

Apache adalah perangkat lunak sumber terbuka yang menjadi alternatif dari *Server web Netscape*. *Server HTTP Apache* atau *Server Web* atau *WWW Apache* merupakan *Server web* yang dapat dijalankan di banyak sistem operasi yang berguna untuk melayani dan memfungsikan situs *web*. *Apache* dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang dibawah naungan *Apache Software Foundation*.

2. *MySQL*

MySQL adalah singkatan “*My Structured Query Language*”. Program ini berjalan sebagai *Server* menyediakan *multi-Users* mengakses ke sejumlah *Database*. *MySQL* umumnya digunakan oleh perangkat lunak bebas yang memerlukan fitur penuh sistem manajemen *Database*, seperti *WordPress*, *PHPBB* dan perangkat lunak lain yang dibangun pada perangkat lunak *LAMP*. Ia juga digunakan dalam skala sangat tinggi *World Wide Web*, termasuk produk-produk *Google* dan *Facebook*.

3. *PHP*

PHP adalah bahasa pemrograman *script* yang banyak dipakai untuk memrogram situs *web dinamis*, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain. Contoh terkenal dari aplikasi *PHP* adalah *PHPBB* dan *MediaWiki* (*software* di belakang *Wikipedia*).

Bagian-Bagian XAMPP:

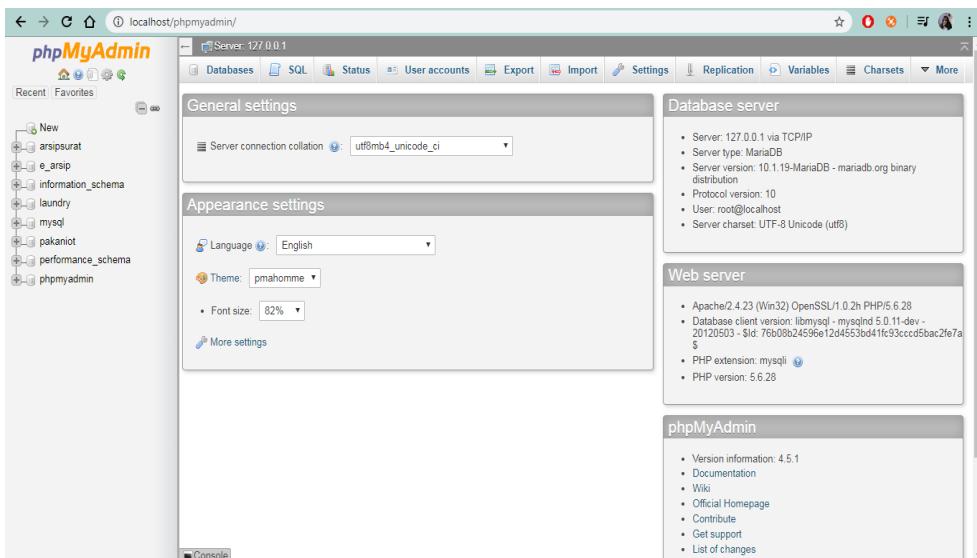
1. *Htdocs*

Name	Date modified	Type	Size
anonymous	15/12/2013 13:52	File folder	
apache	15/12/2013 13:52	File folder	
cgi-bin	15/12/2013 13:56	File folder	
contrib	15/12/2013 13:52	File folder	
FileZillaFTP	15/12/2013 13:56	File folder	
htdocs	15/12/2013 16:08	File folder	
img	15/12/2013 13:52	File folder	
install	15/12/2013 13:56	File folder	
licenses	15/12/2013 13:52	File folder	

Gambar 3. 4 Contoh *Htdocs*

Htdocs adalah sebuah *folder* yang digunakan sebagai tempat penyimpanan berkas seperti *PHP*, *HTML*, dan *script* lain yang digunakan dalam sebuah halaman *Website*. Secara kapasitas penyimpanan, *XAMPP* tergantung dari seberapa besar kapasitas *hardisk* di laptop atau komputer anda. Sedangkan bila menggunakan *hosting online*, maka tergantung pilihan waktu membeli sebuah *hosting*.

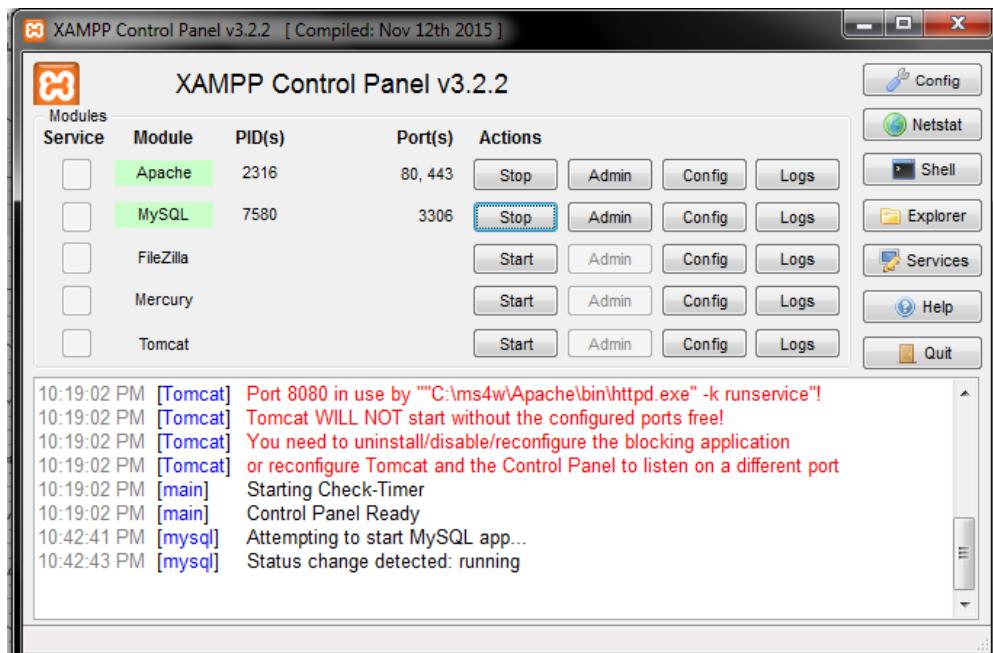
2. *PHPMyAdmin*



Gambar 3. 5 Contoh *PHPMyAdmin*

PHPMyAdmin adalah sebuah tempat yang digunakan untuk mengelola *Database MySQL* yang berada di komputer atau laptop. Untuk mengakses *PHPMyAdmin* yakni dengan membuka *Browser internet* (*Mozilla* atau *Chrome*) lalu ketikkan alamat <http://localhost/PHPMyAdmin>, maka akan muncul tampilannya.

3. Control Panel



Gambar 3. 6 Contoh *Control Panel*

Control Panel adalah sebuah layanan untuk mengelola XAMPP baik itu mengontrol (*start* atau *stop* XAMPP) serta layanan *service* lainnya. Secara *online* di dalam *hosting* atau *VPS* dikenal *CPanel*.

Kelebihan dari XAMPP, diantaranya:

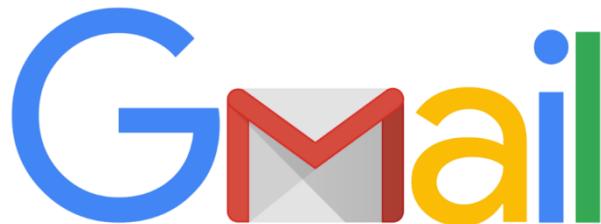
1. *Database Storage Engine* ini banyak digunakan oleh *programmer* apalagi oleh *web developer* karena sifatnya yang *free*. Untuk yang *expert* sudah ada yang bayar.
2. Kemampuannya sudah bisa diandalkan, mempunyai kapasitas yang cukup mumpuni sekitar 60.000 tabel dengan jumlah *record* mencapai 5.000.000.000 bahkan untuk yang terbaru sudah lebih.

3. Keamanan datanya cukup aman walaupun tidak sehebat *Postgre* apalagi *Oracle*.
4. *Engine* ini *multiplatform* sehingga mampu diaplikasikan di berbagai sistem operasi. *MySQL* cocok diaplikasikan di aplikasi kelas kecil dan menengah.
5. Kelebihan paling utama *engine* ini adalah kecepatannya.

Kekurangan dari *XAMPP*, diantaranya:

1. Tidak cocok untuk menangani data dengan jumlah yang besar, baik untuk menyimpan data maupun untuk memproses data.
2. Memiliki keterbatasan kemampuan kinerja pada *Server* ketika data yang disimpan telah melebihi batas maksimal kemampuan daya tampung *Server* karena tidak menerapkan konsep *Technology Cluste*.

3.1.3 Email



Gambar 3. 7 Ilustrasi *E-mail*

3.1.3.1 Apa itu *E-mail* ?

E-mail atau surat elektronik adalah suatu alat atau sarana untuk mengirim dan menerima surat atau pesan dengan format dalam bentuk digital melalui jalur jaringan komputer dan internet.

3.1.3.2 Elemen-Elemen *E-mail*

Menurut David Alex Lamb, dalam proses pengiriman dan penerimaan *email* terdapat beberapa elemen yang sangat berpengaruh terhadap proses tersebut. Elemen-elemen yang dimaksud adalah :

1. *Sender*, merupakan orang yang menyusun dan mengirimkan *email*.
2. *Mail agent*, merupakan perangkat lunak yang digunakan oleh pengirim untuk melakukan penyusunan *email*.
3. *Message*, merupakan representasi komputer dari apa yang ingin disampaikan oleh pengirim.

4. *Email transport subsystem*, merupakan sebuah sistem yang menangani transportasi *email*.
5. *Receiver*, merupakan tujuan atau orang yang menerima *email*.
6. *Email agent software*, merupakan sebuah perangkat lunak yang digunakan untuk membaca *email*.
7. *Email address*, merupakan sekumpulan karakter yang digunakan untuk mengenali pengirim dan penerima.

3.1.3.3 Perbandingan *E-mail* Dengan Surat Biasa

Pada dasarnya sistem pengiriman dan penerimaan *E-mail* sama seperti pada sistem penerimaan dan pengiriman surat biasa. Berikut merupakan perbandingan antara *e-mail* dan surat biasa.

Tabel 3. 1 Perbandingan *E-mail* dengan surat biasa

Pembanding	<i>E-mail</i>	Surat Biasa
<i>Mail Agent</i>	<i>Outlook Express</i>	Pena/Mesin Tik
Media Pesan	<i>Text, HTML</i>	Kertas, Amplop
<i>Subsystem</i>	<i>SMTP, POP3, IMAP</i>	POS, <i>DHL</i> , dan jasa kurir lainnya
Alamat	<i>E-mail</i>	Alamat Rumah

Dalam sistem *E-mail* terdapat dua buah *subSystem E-mail* .

1. MUA (*Mail Users Agent*) merupakan sebuah program di komputer lokal yang mendukung penggunaan *command-based* (berbasis perintah), *Menu based* (berdasarkan *Menu*) atau grafikal. MUA digunakan untuk membaca dan mengirimkan *E-mail* . Contoh aplikasi dari MUA antara lain *Outlook Express, KMail, Eudora, Pine*.

- Subsistem kedua adalah MTA (*Mail Transport Agent*) merupakan sebuah *daemon System* yang berjalan di belakang (*background*) dalam proses pemindahan *E-mail* dimana fungsi MTA adalah sebagai pengatur transportasi *E-mail* dari pengirim ke penerima.

3.1.3.4 Format *E-mail*

Sebuah pesan merupakan sebuah kumpulan karakter yang dipisahkan ke dalam beberapa baris karakter. Baris karakter yang digunakan dibatasi oleh dua buah karakter berupa *Carriage Return* (CR) dengan nilai ASCII 13, yang kemudian diikuti langsung oleh *Line Feed* (LF) dengan nilai karakter ASCII 10 (RFC 2822).

E-mail terdiri dari dua bagian utama, *header* dan *body*. Bagian *header* minimal terdiri dari tiga isian utama berupa “*date*”, “*from* : “, dan “*to*”, dimana setiap *field* pada *header* memiliki aturan penulisan sendiri.

Beberapa *field* yang umum digunakan pada bagian *header*:

Tabel 3. 2 Field header pada *E-mail*

<i>Header Field</i>	Berisi
<i>To:</i>	Satu atau lebih alamat <i>email</i> penerima primer
<i>CC:</i>	Satu atau lebih alamat <i>email</i> penerima sekunder
<i>BCC:</i>	Satu atau lebih alamat <i>email</i> penerima <i>blind carbon copy</i>
<i>From:</i>	Orang yang membuat <i>email</i>
<i>Subject:</i>	Judul <i>email</i>
<i>Sender:</i>	Alamat <i>email</i> pengirim

<i>Received:</i>	Isian yang ditambahkan setiap melalui MTA
<i>Return Path:</i>	Jalur kembali mail ke pengirim

3.1.3.4 Tahapan Proses Pengiriman *E-mail*

1. Pengiriman pesan

Pada proses ini, pengirim menyusun pesan dan mengirimkannya dengan *email agent*, yang akan memberikan perintah kepada sistem *transport email* untuk mengantarkan pesan tersebut ke tujuan.

a. *CompOSIng*

Penyusunan *E-mail* mirip dengan penyusunan naskah, dimana *E-mail agent* pada umumnya memiliki beberapa editor-editor teks. Yang membedakan adalah format *E-mail* yang terdiri dari dua jenis, *header* dan *body*. Mengacu pada RFC 822, *header* merupakan bagian *E-mail* yang terdiri dari alamat, judul dan tembusan. Sedangkan *body* merupakan bagian dari *E-mail* yang memuat pesan yang akan disampaikan oleh pembuat *E-mail* berupa teks murni atau disertakan *File* dengan format tertentu yang nantinya dikodekan sebagai teks yang disebut dengan *MIME (Multipurpose Internet Mail Extensions)*. *E-mail agent* biasanya menyediakan ruang khusus sebagai tempat menaruh konsep atau rancangan *E-mail* yang sudah dibuat sehingga pengguna bisa melakukan *editing* kembali.

b. Pengantrian dan pengiriman

Setelah *E-mail* disusun dan kemudian pengguna memutuskan untuk mengirimkan pesan tersebut dengan menggunakan perintah tertentu, maka *E-mail agent* akan memeriksa alamat tujuan (sebagai contoh *header field to:*) sesuai dengan aturan penulisan alamat *E-mail*

tujuan atau belum. Jika penulisan salah maka *E-mail agent* akan memberikan pemberitahuan kepada pengguna untuk memeriksa ulang dan memperbaiki. Sebaliknya jika penulisan alamat *E-mail tujuan* sudah benar, maka *E-mail* akan dikirimkan. Dimana *E-mail agent* akan memanggil *E-mail transport subSystem* untuk mengirimkan *E-mail*.

Beberapa pengguna koneksi internet *dial up* menyusun *E-mail* pada saat *off line* dan baru akan mengirimkannya pada saat *online*. Kasus lain terjadi ketika suatu *E-mail* tidak bisa dikirimkan karena ada *E-mail* lain yang memiliki prioritas lebih tinggi. *Mail transport* dan *mail agent* memiliki tempat antrian khusus yang digunakan untuk menampung *email* yang akan dikirimkan, jika kondisi memungkinkan maka *E-mail -E-mail* yang ditampung tersebut akan dikirim ke komputer tujuan.

2. Pemindahan

a. Pengalamanan *E-mail*

Satu atau lebih alamat tujuan oleh *mail transport System* digunakan untuk mengetahui kemana *E-mail* akan dikirimkan. Alamat *E-mail* yang digunakan berupa sekumpulan teks yang mengidentifikasi keberadaan kotak *E-mail* (*mailbox*) pada IP atau *domain* tertentu.

Layanan *transport* mengirimkan kumpulan teks *site* pada *Domain Name System* (DNS) yang akan mengubahnya ke dalam alamat IP kemudian menghubungkan ke alamat IP yang diberikan dan meminta komputer tujuan untuk menerima dan mengirimkannya pada *mailbox* yang dituju.

b. *E-mail Server*

Ketika *email* dikirimkan dan sebelum sampai kepada *mailbox* yang dituju, terlebih dahulu isi pesan *email* tersebut diproses pada *E-mail Server*. Ada tiga hal yang dilakukan *E-mail Server* sebagai respon dari permintaan tersebut, berupa :

- i. Menerima pesan dan menyampaikannya dalam *mailbox* tujuan,
- ii. Menerima pesan ke alamat lain yang telah ditetapkan oleh pemilik *mailbox*,
- iii. Menolak pesan dan memberikan pemberitahuan bahwa pesan tidak terkirim yang diakibatkan karena *mailbox* tujuan tidak tersedia, *mailbox* tujuan dalam kondisi penuh atau karena terdapat kerusakan pada *Server*.

Dalam penggunaannya terdapat dua buah aplikasi *mail Server*, kedua aplikasi tersebut berupa :

i. SMTP

Protokol yang digunakan untuk mengelola lalu lintas *email* keluar masuk suatu jaringan.

ii. POP

Protokol yang digunakan untuk mengambil *email* dari tempat penampungan *E-mail* pada *email Server*.

3. Penerimaan *E-mail*

Suatu saat *mail agent* melakukan pengecekan secara otomatis atau berdasarkan permintaan pengguna, apakah ada *E-mail* masuk ke dalam *mailbox* pada *E-mail Server*. Jika terdapat *email* masuk maka kemudian *mail agent* tersebut akan menyimpan *E-mail* tersebut dalam *Database mail agent* tersebut.

3.1.3.5 Layanan *E-mail*

Terdapat lima layanan dasar yang didukung oleh sistem *E-mail*. Menurut Tanenbaum, kelima dasar tersebut adalah :

1. KompOSIsi

Berhubungan dengan proses pembuatan pesan dan/atau balasan *email*, pengolah teks digunakan untuk badan *email* dan sistem membantu pengalaman dan beberapa isian pada bagian kepala (*header*) *email*.

2. Pengiriman

Ketika seorang pengirim melakukan proses pengiriman sebuah pesan kepada penerima maka ketika itu juga diperlukan sebuah hubungan dari pengirim kepada penerima, atau media lain yang berada diantaranya. Setelah koneksi dilakukan, pemutusan koneksi dilakukan jika *email* telah dikirimkan. Pengiriman ini merupakan proses *background* pada sistem.

3. Pelaporan

Memberikan laporan kepada pengirim terhadap kejadian yang dialami oleh pesan. Kondisi yang ditemukan berupa laporan keberhasilan pengiriman pesan selain itu bisa juga pelaporan berupa penolakan atau hilangnya pesan yang dikirim.

4. Tampilan

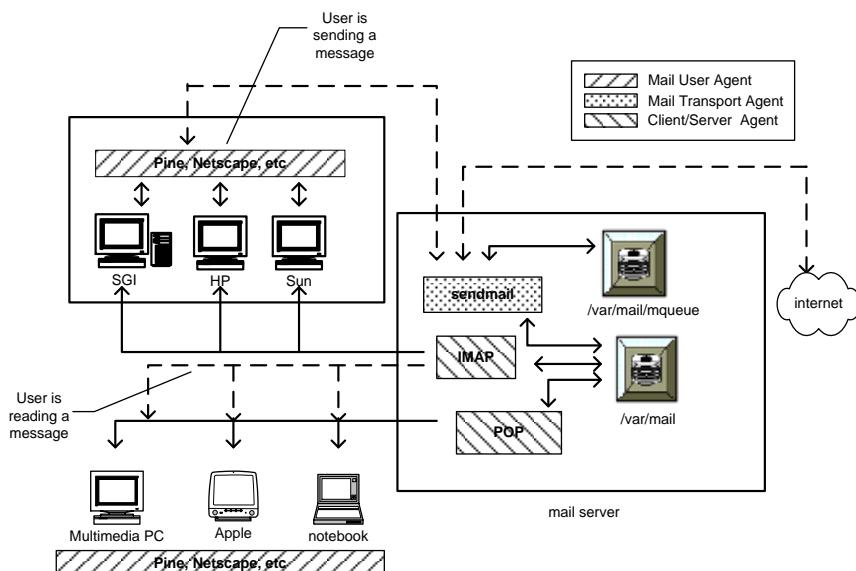
Format tampilan pada layar komputer penerima sesuai dengan format yang telah dibuat oleh pengirim. Meskipun pada kenyataannya terdapat perbedaan format MUA antara penerima dan pengirim.

5. Penempatan

Berhubungan dengan apa yang akan dilakukan penerima setelah menerima *E-mail*.

3.1.3.6 Komponen *E-mail*

Sebuah *email* secara umum memiliki arsitektur seperti yang terlihat pada gambar 3.8, dengan sejumlah komponen penyusun. Pesan yang dikirimkan akan tersimpan di dalam *email Server* yang dikirimkan melalui *mail Users agent*. *Mail Users agent* mengambil dan mengirimkan pesan tersebut ke dan dari tempat penyimpanan melalui protokol internet yang dapat diandalkan.



Gambar 3. 8 Arsitektur *E-mail Client/Server*

komponen tersebut adalah :

1. *Mail Users Agent*

Merupakan program yang digunakan untuk membuat dan membaca *email*. MUA bisa disebut juga sebagai *email reader* yang dapat

menerima bermacam - macam perintah untuk pembuatan, penerimaan dan penjawaban pesan. Selain itu MUA juga dapat digunakan untuk memanipulasi *mailbox email* pengguna.

Beberapa MUA mengizinkan pemakaian format *Multipurpose Internet Mail Extension* (MIME) yang dapat digunakan untuk melampirkan *File* ke dalam suatu pesan. Dengan MIME ini, pesan yang dikirimkan dapat melampirkan *File-File* lain. *MailX*, *pine* dan *netscape* merupakan beberapa contoh MUA.

2. *Mail Transport Agent*

Pada saat proses pengiriman *email*, *email* tersebut diberikan ke MTA untuk dilakukan proses yang lebih lanjut.

Fungsi dari MTA adalah sebagai berikut :

- a. MTA menggunakan alamat tujuan untuk menentukan bagaimana pesan tersebut harus dikirimkan.
- b. MTA dapat menggunakan *aliases/daftar distribusi* untuk mengirimkan salinan dari sebuah pesan ke berbagai tujuan.
- c. MTA menerima dan memproses *email* yang masuk dari mesin lain dalam jaringan.

3. *Mail Channel* dan *Delivery Agent*

Mail channel memiliki dua komponen utama yaitu :

- a. Tabel yang menentukan saluran yang akan digunakan untuk mengirimkan pesan *email*.
- b. *Delivery agent* yang akan melakukan pengiriman pesan bagi saluran yang telah ditentukan.

MTA akan mengenali saluran pengiriman pertama yang sesuai dengan alamat pesan pada tabel dan menggunakan agen pengiriman

yang sesuai untuk mengirimkan pesan. Alamat yang tidak sesuai dengan saluran yang ada akan dianggap sebagai alamat yang tidak dapat dikirimkan dan akan dikembalikan ke pengirim. Jenis-jenis saluran yang digunakan :

- a. *Local channel* yang menangani pengiriman *email* ke pengguna lokal.
- b. *SMTP channel* akan mengirimkan *email* melalui jaringan TCP/IP dengan menggunakan DNS dan SMTP.
- c. *Badhost channel* yang secara default akan mengantrikan pesan-pesan yang sementara waktu tidak dapat dikirimkan karena nama *Server* tidak siap.

Pesan *email* akan *di-route* dari pengirim ke penerima oleh MTA pada *host email* yang berbeda. Komunikasi diantara *host* yang berbeda ditangani oleh program yang disebut *mailer*.

Beberapa *mailer* yang biasa digunakan adalah :

- a. *Local* : pengiriman *local*
- b. *Ether* : pengiriman *Ethernet*
- c. *Ddn* : pengiriman *internet*
- d. *Uucp* : pengiriman *uucp*
- e. *Error* : *mailer* yang digunakan untuk menghasilkan pesan *error* ke pengguna.

3.1.3.7 Multipurpose Internet Mail Extension

Pada awalnya *email* ditulis dalam bahasa Inggris dengan menggunakan format ASCII, namun seiring dengan perkembangan zaman kebutuhan akan *email* sudah merambah tidak hanya oleh pengguna yang berbasiskan ASCII saja. Untuk itu diperlukan sebuah cara yang dapat merepresentasikan suatu

huruf lain dalam sebuah *email*. RFC 2045 mendefinisikan ulang format pesan MIME ditujukan untuk menangani:

1. Pesan yang digunakan bukan dalam bentuk US –ASCII.
2. Sekumpulan bentuk pesan yang bukan teks.
3. Pesan dengan sifat *Multi-part*.
4. Informasi *header mail* yang digunakan bukan dalam karakter US-ASCII.

Selain itu permasalahan pengiriman *email* yang bisa diatasi dengan MIME berupa :

1. Pesan dalam aksara aksentuasi (contoh : Prancis)
2. Pesan dalam aksara *non latin* (contoh : Rusia)
3. Pesan dalam bentuk aksara tanpa alfabet (contoh : Jepang)
4. Pesan dalam bentuk *non-Text* (contoh : gambar dan video)

Tabel 3. 3 *Header tambahan MIME*

Header	Fungsi
<i>MIME – Version</i> :	Identifikasi versi MIME
<i>Content-description</i> :	Memberitahukan apa yang ada dalam pesan
<i>Content-ID</i> :	Pengenal khusus / unik
<i>Content-Transfer-Encoding</i> :	Membungkus pesan untuk pengiriman
<i>Content-Type</i> :	Memberitahukan <i>Type</i> pesan

MIME didefinisikan dengan lima *header* baru. *Header MIME Text* digunakan untuk memberitahukan *mail Users agent* apakah pesan berupa *plainText* atau bukan. Jika pesan tidak memiliki “*MIME-Version*:

“pesan tersebut diasumsikan dalam format *plainText* dan beraksara inggris. *Header “Content-Description:*” berisi pesan ASCII, berfungsi untuk memberitahukan apa yang ada dalam pesan kepada si penerima *email* tersebut, sehingga penerima pesan bisa memutuskan apakah pesan tersebut akan dibaca, dikodekan atau dibuang.

Header “Content-ID:” merupakan identitas dari MIME yang bersifat unik. *Header “Content-Transfer-Encoding:*” digunakan untuk menentukan bagaimana pesan dikemas untuk pengirimannya melalui suatu jaringan. Untuk data biner pada umumnya menggunakan *base64* dan *quoted printable encoding*. Jika pesan lebih banyak berupa kode ASCII daripada *non ASCII* maka yang digunakan adalah *quoted printable encoding*, karena lebih efisien dibandingkan menggunakan *base64*. Sedangkan “*Content-Type:*” mengidentifikasi sifat atau karakter dari pesan.

Tabel 3. 4 Konten *Type*

Tipe	Subtipe	Deskripsi
<i>Text</i>	<i>Plain</i>	Teks murni atau <i>unformatted Text</i> .
	<i>RichText</i>	Teks dengan perintah pemformatan sederhana.
<i>Image</i>	GIF	Gambar dalam bentuk GIF.
	JPEG	Gambar dalam bentuk JPEG.
<i>Audio</i>	<i>Basic</i>	<i>Audible Sound</i> .
<i>Video</i>	MPEG	Film dan format MPEG.
<i>Application</i>	<i>Octet-Stream</i>	Rangkaian byte yang belum diterjemahkan (<i>uninterpreted</i>).
	<i>Postscript</i>	Dokumen <i>printable</i> dalam format Postscript.

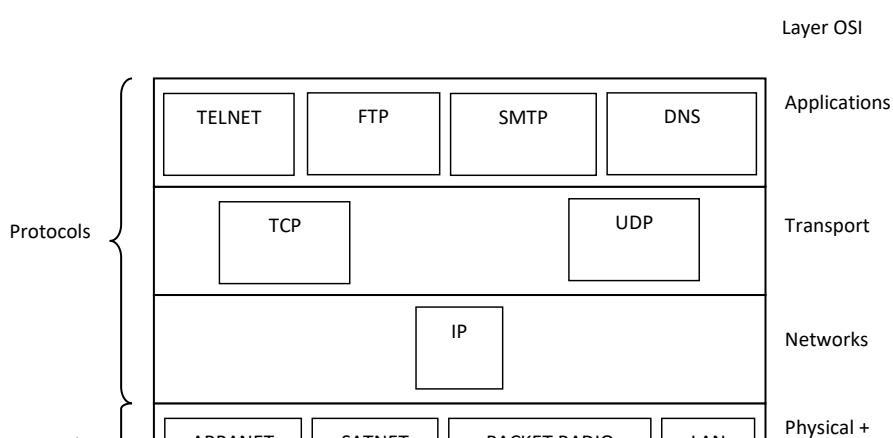
<i>Message</i>	RFC822	Pesan dalam bentuk MIME RFC 822.
	<i>Partial</i>	Pesan dipecah untuk kepentingan pengiriman.
	<i>External Body</i>	Pesan diperoleh dari jaringan internet.
<i>Multiport</i>	<i>Mixed</i>	Mengijinkan setiap bagian pesan memiliki tipe berbeda.
	<i>Alternative</i>	Pesan yang sama dalam format yang berbeda.
	<i>Parallel</i>	Semua bagian pesan ditampilkan secara simultan.
	<i>Digest</i>	Setiap bagian <i>email</i> dalam bentuk pesan RFC 822 komplet.

3.1.4 Protokol *E-mail*

Simple Mail Transport Protocol

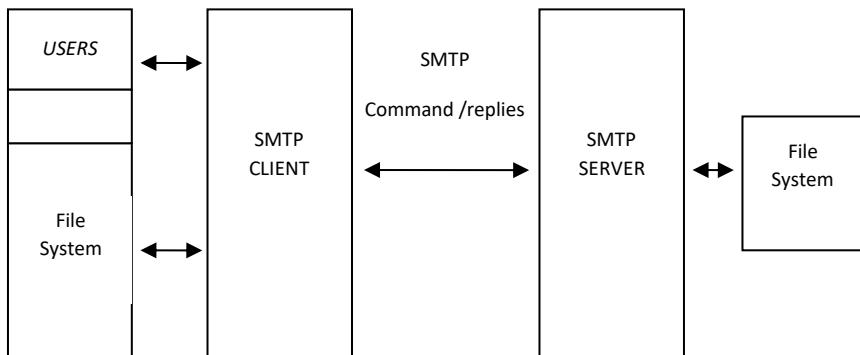
Seperti yang sudah dibahas sebelumnya SMTP merupakan protokol yang digunakan untuk mengelola lalu lintas *email* keluar masuk suatu jaringan.

SMTP berada pada *layer* aplikasi dengan *port* 25. Fungsi utama dari SMTP adalah menyampaikan pesan dari komputer pengirim ke komputer penerima, baik dalam jaringan yang sama (LAN atau WAN) maupun ke komputer penerima dalam suatu jaringan yang berbeda.



Gambar 3. 9 Layer *OSI*

Model SMTP yang dijelaskan pada RFC 2821 berupa :



Gambar 3. 10 Model STMP

Jika akan mengirimkan suatu *email*, maka *SMTP Client* akan membuka kanal dua arah ke *SMTP Server*. Dalam hal ini *SMTP Server* bisa merupakan tujuan akhir, namun kadang bisa juga menjadi perantara antara komputer penerima dengan komputer pengirim atau berupa gerbang yang menghubungkan komunikasi SMTP dengan protokol lain.

Koneksi *SMTP Client-Server* diawali dengan proses inisialisasi, *SMTP Server* akan memberikan status bisa digunakan atau tidak. Jika tidak bisa digunakan maka koneksi diputus dan jika statusnya bisa digunakan *SMTP Client* bisa memulai pengiriman kumpulan perintah yang diperlukan seperti menentukan alamat pengirim, alamat tujuan, serta pesan yang akan disampaikan. Setelah pesan dikirimkan oleh *SMTP Server*, *SMTP Client* bisa meminta koneksi diputus atau dimulai untuk pengiriman *email* lainnya.

1. *Internet Message Access Protocol dan Post Office Protocol*

Protokol IMAP dan POP digunakan untuk dapat menjembatani *email Users* dan *Server*. Hal ini dikarenakan kondisi *email Users* yang digunakan tidak secara terus menerus terkoneksi sehingga *email* yang masuk akan ditampung pada *email Server*. Kemudian jika akan dibaca, *email* tersebut didownload oleh pengguna setelah terkoneksi dan berinteraksi dengan *email Server*.

a. *Post Office Protocol*

POP merupakan protokol yang digunakan untuk mengambil pesan dari *mailbox* pada komputer *Server* dan menyimpannya pada komputer lokal pengguna POP3. *Server* menggunakan *port* 110 pada TCP/IP. Jika ada *Client* yang akan menggunakan layanan *Server*, maka koneksi antara keduanya dilangsungkan. Setelah terkoneksi, *Server* POP3 akan memberikan sebuah pesan sambutan yang kemudian dilanjutkan pada tahap berikutnya yaitu tahapan otorisasi, dimana *Client* harus mengidentifikasi dirinya ke *Server* POP3 dengan mengirimkan *Users id* dan *Password*.

Jika otorisasi berhasil dan sesuai dengan data yang tersedia di *Server*, maka *Server* akan mengambil data yang dibutuhkan dalam koneksi tersebut dan dilanjutkan dengan tahapan transaksi.

Pada tahapan transaksi, pengguna bisa menggunakan beberapa perintah untuk berinteraksi dengan *Server*, semisal menampilkan daftar *email* yang tersedia dalam *mailbox*. Semua pesan yang dikirimkan dalam koneksi POP3 berupa kode ASCII dan format pesan *email* yang dikirimkan diasumsikan sesuai dengan standar pada RFC 822. Karena semua pesan tidak terenkripsi jika dilakukan penyadapan, maka pesan yang dikomunikasikan selama koneksi dapat dibaca langsung. Solusi untuk permasalahan ini dapat dibangun

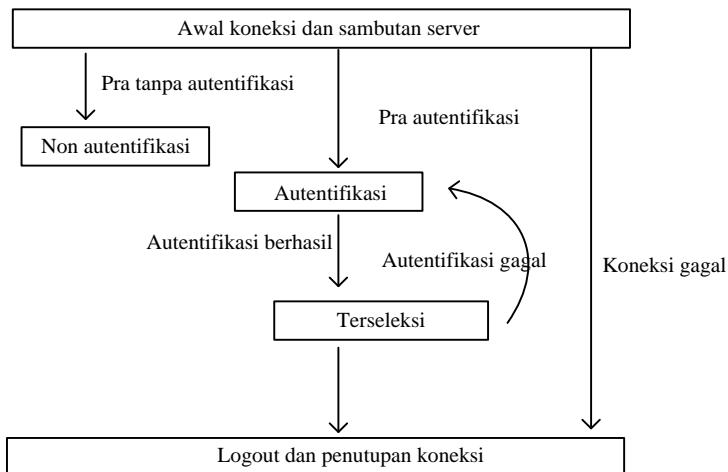
dengan koneksi SSH, menggunakan otentikasi/enkripsi (PGP atau PEM) pada pesan *email*.

b. *Internet Message Access Protocol*

IMAP merupakan sebuah protokol yang dirancang agar *Users* dapat mengakses *email* pada *mailbox* serta dapat berinteraksi dengan *mail Server*. *Port* yang digunakan oleh protokol ini dalam TCP/IP adalah *port* nomor 143.

Sesi koneksi IMAP terdiri dari koneksi *Client – Server*, pesan sambutan dan interaksi *Client – Server*. Interaksi *Client – Server* terbagi menjadi perintah *Client*, *data Server* dan tanggapan penyelesaian perintah *Server*. Koneksi IMAP terbagi menjadi empat kondisi, yaitu :

- Non-autentifikasi, dilakukan agar pengguna mendapatkan hak untuk menjalankan perintah dalam koneksinya. Jika tidak bisa dilakukan, pengguna hanya bisa memberikan perintah yang bisa dijalankan pada semua keadaan berupa *Capability*, *Noop* dan *Logout*.
- Terautentifikasi, pada jenis ini pengguna diberikan hak akses dan pengiriman perintah. Untuk dapat melakukan ini *Users* harus memiliki *mailbox* untuk diakses sesuai dengan wewenangnya sebelum perintah-perintah yang dapat mempengaruhi pesan dikirimkan ke *Server*.
- Terseleksi, keadaan ini dimulai setelah *mailbox* yang ditentukan dapat diakses oleh *Users*.
- *Logout* (keluar), kondisi ini akan memutuskan koneksi. Kondisi ini terjadi karena pemutusan koneksi yang disengaja oleh *Users*, selain itu bisa juga diakibatkan karena sesuatu hal semisal tidak terjadinya interaksi dalam waktu tertentu.



Gambar 3. 11 Koneksi *Client – Server* IMAP

3.2 BAHASA PEMOGRAMAN

3.2.1 CSS

CSS (*Cascading Style Sheet*) adalah salah satu bahasa desain *web (style sheet language)* yang mengontrol *format* tampilan sebuah halaman *web* yang ditulis dengan menggunakan penanda (*markup language*). Biasanya CSS digunakan untuk mendesain sebuah halaman *HTML* dan *XHTML*, tetapi sekarang CSS bisa diaplikasikan untuk segala dokumen *XML*, termasuk *SVG* dan *XUL* bahkan *ANDROID*. CSS dibuat untuk memisahkan konten utama dengan tampilan dokumen yang meliputi *layout*, warna dan *font*. Pemisahan ini dapat meningkatkan daya akses konten pada *web*, menyediakan lebih banyak fleksibilitas dan kontrol dalam spesifikasi dari sebuah karakteristik dari sebuah tampilan, memungkinkan untuk membagi halaman untuk sebuah *formatting* dan mengurangi kerumitan dalam penulisan kode dan struktur dari konten, contohnya teknik *tableless* pada desain *web*.

CSS juga memungkinkan sebuah halaman untuk ditampilkan dalam berbagai *style* dengan menggunakan metode pembawaan yang berbeda pula, seperti *on-screen*, *in-print*, *by voice*, dan lain-lain. Sementara itu, pemilik konten *web* bisa menentukan link yang menghubungkan konten dengan *File CSS*. Tujuan utama *CSS* diciptakan untuk membedakan konten dari dokumen dan dari tampilan dokumen, dengan itu, pembuatan ataupun pemrograman ulang *web* akan lebih mudah dilakukan. Hal yang termasuk dalam desain *web* diantaranya adalah warna, ukuran dan *formatting*. Dengan adanya *CSS*, konten dan *desain web* akan mudah dibedakan, jadi memungkinkan untuk melakukan pengulangan pada tampilan-tampilan tertentu dalam suatu *web*, sehingga akan memudahkan dalam membuat halaman *web* yang banyak, yang pada akhirnya dapat memangkas waktu pembuatan *web*. Fungsi utama *CSS* adalah merancang, merubah, mendisain, membentuk halaman *wesite* (blog juga *Website*). dan isi dari halaman *Website* adalah tag-tag *html*, logikanya *CSS* itu dapat merubah tag-tag *html* (yang sederhana) sehingga menjadi lebih fungsional dan menarik.

Untuk cara kerjanya *CSS* beroperasi melalui tag <style> dengan atribut class warna. Dengan adanya *CSS* pada *HTML* tersebut maka pengaturan warna teks akan menjadi lebih mudah. Saat kamu ingin mengganti warna teks cukup mengetikkan tag tanpa harus *Menulis* ulang perintah. Jadi bisa disimpulkan bahwa *CSS* akan menghemat waktumu dengan perintah-perintah yang efisien. Hal ini bisa terjadi karena *CSS* sendiri dikembangkan untuk bisa mengubah tampilan laman *Website* tanpa harus mengganti isi konten. Jika kembali pada perumpamaan manusia dan pakaian di poin sebelumnya, dengan *CSS* kamu tidak mengubah bentuk manusianya tapi hanya mengganti pakaiannya. Dengan begitu untuk mengubah dan memprogram ulang tampilan *Website* pun bisa dilakukan dalam waktu cepat. peran *CSS* untuk *Website* sangatlah penting. Tanpa adanya *CSS*, tampilan *Website* akan membosankan

atau bahkan membutuhkan waktu lama untuk loading. Bayangkan saja kamu hanya bisa bergantung pada HTML untuk membuat sebuah situs. Bukan cuma tampilan situs akan “hambar” tapi kamu juga butuh waktu lebih lama karena harus berulang kali mengetikkan perintah.

Macam-macam CSS, diantaranya yaitu:

1. *Inline style sheet*

Pertama ada *inline style sheet*. Sederhananya, CSS model ini adalah CSS dengan perintah pemrograman yang letaknya ada pada objek. Misalnya kamu ingin mengubah sebuah tulisan pada laman tertentu di *Website* milikmu, inline style sheet CSS harus menempel pada elemen tulisan tersebut. Kamu cukup menambahkan tag <style> saja untuk menerapkan CSS ini.

2. *External style sheet*

Selain *inline style sheet* ada *external style sheet*. CSS ini letaknya berbeda dengan laman yang akan diubah. Cara ini lebih praktis daripada *inline style sheet* karena bisa menghemat ruang dan bisa digunakan berulang-ulang untuk laman *web* yang berbeda. Kamu bisa mengenali CSS tipe ini lewat tag <link rel>. Tag ini akan menghubungkan halaman *coding* pada *external style sheet CSS* yang terpisah.

3. *Embedded style sheet*

Terakhir ada *embedded style sheet*. CSS model ini sama seperti *inline style sheet*, sama-sama berada pada satu laman *coding*. Maka tidak mengherankan jika *embedded style sheet* terkadang disebut dengan *internal style sheet*. Biasanya CSS ini diapit oleh tag <head> </head>

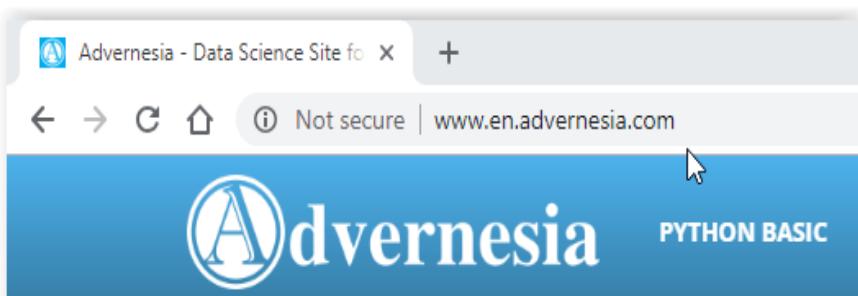
dan diawali dengan tag <style>. *Embedded style sheet* sering digunakan untuk mengatur laman *web* dengan tampilan yang unik. Misalnya dalam satu paragraf tulisan ada kalimat yang berbeda dan hal tersebut terus berulang.

3.2.2 HTTP (*HyperText Transfer Protocol*) dan HTTPS (*Hyper Text Transfer Protocol Secure*)

1. HTTP (*HyperText Transfer Protocol*)

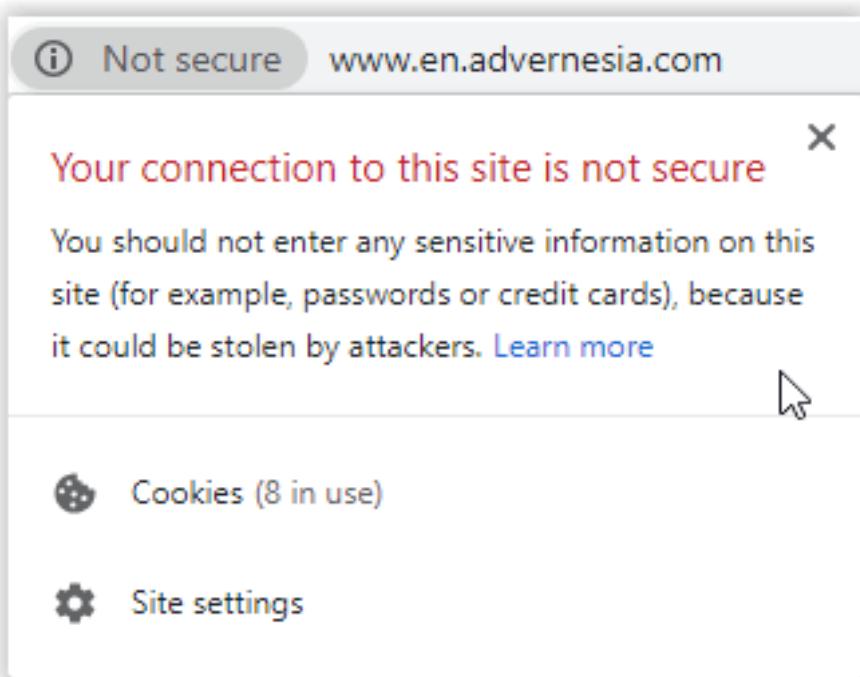
HTTP kepanjangan dari *Hyper Text Transfer Protocol*, adalah protokol standar yang digunakan sebuah *Website* untuk melakukan transfer data antar komputer *Server* (misalnya *Server* hosting) dengan komputer *Client* (komputer yang mengakses *Website*). *HTTP* memegang peranan yang penting untuk mengatur aliran data dari komputer *Server* terkait data apa saja yang akan diberikan kepada komputer *Client* dan memberikan instruksi kepada komputer *Server* untuk merespon komunikasi dari komputer *Client*.

Semua *Website* menggunakan protokol *HTTP*. Pada umumnya *Browser* tidak menampilkan protokol *HTTP* yang digunakan, melainkan status koneksi *Website* tersebut. *Browser* akan menampilkan status “*Not Secure*” atau “**Tidak aman**” untuk protokol *HTTP*.



Gambar 3. 12 Contoh situs *http: Browser Google Chrome* menampilkan status *Website* dengan protokol *HTTP*

Status “Not Secure” disebabkan karena pengelola Website hanya menggunakan protokol HTTP saja tanpa memberikan perlindungan keamanan data pada Website tersebut.



Gambar 3. 13 Status “Not Secure” pada Browser Google Chrome

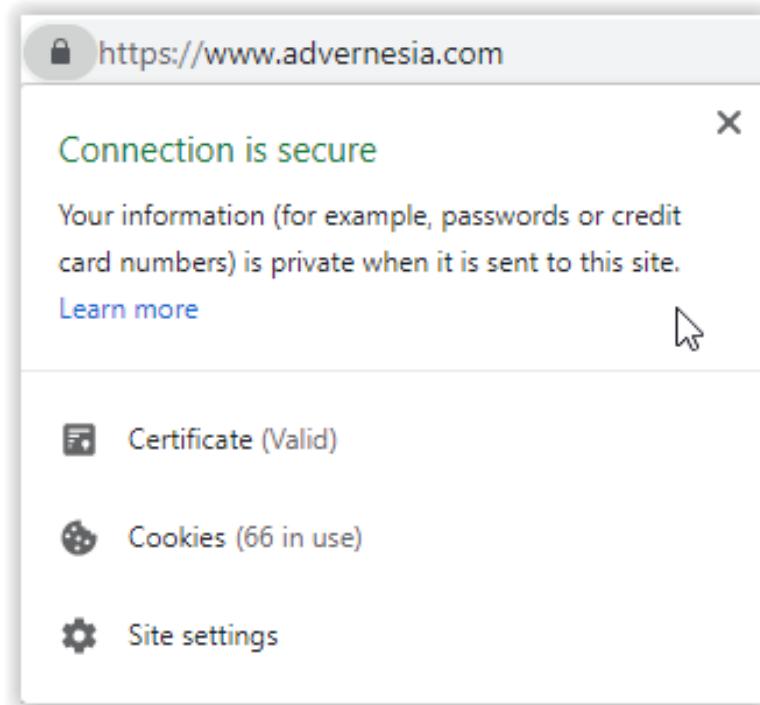
Status tersebut memperingati pengguna Website untuk berhati-hati ketika memasukkan data pribadi, akun, *Password*, hingga kartu kredit pada Website bersangkutan. Hal ini disebabkan karena protokol HTTP sangat mudah untuk diretas.

3.2.3 *HTTPS (Hyper Text Transfer Protocol Secure)*

HTTPS kepanjangan *Hyper Text Transfer Protocol Secure*, merupakan protokol *HTTP* yang dilengkapi dengan sistem keamanan (*security*) berupa *SSL (Secure Socket Layer)*. Ibarat kabel telepon, *SSL* merupakan lapisan yang

melindungi serat kabel. *HTTPS* memberikan perlindungan aliran data dari serangan peretasan. *HTTPS* dapat melindungi data *Website*, privasi pengguna, akun bank online, akun pengguna *Website* dari tindakan *cybercrime* seperti peretasan dan mengurangi resiko menipu pengguna dengan *web* tipuan (*phising*). Teknologi *HTTPS* sebenarnya sangat dibutuhkan oleh semua *Website*, tidak hanya *Website* yang dapat menyimpan data pengguna seperti toko online, *Website* bank dan sOSial media. *Website* kecil seperti blog pribadi dan profil perusahaan juga berpotensi dari peretasan.

Browser akan menampilkan status “**Connection is Secure**” atau “**Koneksi aman**” saat mengakses suatu *Website* yang menggunakan protokol *HTTPS*.



Gambar 3. 14 Contoh situs *https*: Status “Secure” pada *Browser Google Chrome*

Status “*Connection is Secure*” memberikan informasi bahwa *Website* tersebut dapat melindungi data yang dimasukkan pengguna *Website* baik akun, *Password*, atau kartu kredit.

3.2.4 HTML (*Hyper Text Markup Language*)

1. Pengenalan HTML

HTML (Hyper Text Markup Language) adalah merupakan sebuah dasar ataupun pondasi bahasa pemrograman sebuah *webpage*, HTML muncul sebagai standar baru dari kemajuan dan berkembangnya internet, pada pertama kali muncul internet masih dalam keadaan berbasis *Text* dimana tampilan sebuah halaman *web* hanya berisikan sebuah *Text* yang monoton tanpa sebuah format dokumen secara visual, bayangkan saja sebuah dokumen *Text* yang dikemas dalam bungkus format seperti tipe *File .txt* atau sering disebut notepad, tanpa paragraph, satu warna, satu ukuran huruf tanpa gambar serta tidak adanya visual format dokumen seperti halnya Ms. Word, hal ini akan sangat membosankan dalam membaca. Dan selain itu pertama kali muncul internet *Users* mengakses masih menggunakan sebuah terminal, hal itu jelas sangatlah tidak friendly. Pemrograman HTML muncul seiring perkembangan teknologi dan informasi.

2. Sejarah HTML

Hyper Text Markup Language (HTML) pertama kali diciptakan dan dikembangkan oleh Tim Berners-Lee pada awal tahun 1990-an yang pada saat itu masih bekerja di CERN. HTML diciptakan dengan tujuan sebagai cara sederhana namun efektif untuk mengkodekan dokumen elektronik. HTML pertama kali dipopulerkan dengan menggunakan *Browser Mosaic*.

Tahun 1980, IBM memikirkan pembuatan suatu dokumen yang akan mengenali setiap elemen dari dokumen dengan suatu tanda tertentu. IBM kemudian mengembangkan suatu jenis bahasa yang menggabungkan teks dengan perintah-perintah pemformatan dokumen. Bahasa ini dinamakan Markup Language, sebuah bahasa yang menggunakan tanda-tanda sebagai basisnya. IBM menamakan sistemnya ini sebagai Generalized Markup Language atau GML.

Tahun 1986, ISO menyatakan bahwa IBM memiliki suatu konsep tentang dokumen yang sangat baik, dan kemudian mengeluarkan suatu publikasi (ISO 8879) yang menyatakan markup language sebagai standar untuk pembuatan dokumen-dokumen. ISO membuat bahasa ini dari GML milik IBM, tetapi memberinya nama lain, yaitu SGML (Standard Generalized Markup Language). ISO dalam publikasinya meyakini bahwa SGML akan sangat berguna untuk pemrosesan informasi teks dan sistem-sistem perkantoran. Tetapi diluar perkiraan ISO, SGML dan terutama subset dari SGML, yaitu HTML juga berguna untuk menjelajahi internet. Khususnya bagi mereka yang menggunakan World Wide Web.

Mulai pada tahun 1989, sebuah nama HTML muncul dari pemikiran Caillau Tim yang bekerja sama dengan Banners Lee Robert yang ketika itu masih bekerja di CERN memulai mengembangkan bahasa pemrograman ini, dan dipopulerkan pertama kali dengan *Browser Mosaic*. Dan mulailah dari tahun 1990 HTML sangat berkembang dengan cepat hingga mencapai versi HTML versi 5.0 yang digarap pada 4 Maret 2010 kemarin oleh W3C.

- Sejarah dari standar HTML :
1. HTML 2.0 (RFC 1866) disetujui sebagai standar 22 September 1995
 2. HTML 3.2 14 Januari 1996
 3. HTML 4.0 18 Desember 1997
 4. HTML 4.01 (minor Fixes) 24 Desember 1999

5. ISO/IEC 15445:2000 (“ISO HTML”, berdasar pada HTML 4.01 Strict) 15 Mei 2000
6. HTML 5 masih dalam draft pengerjaan Januari 2008

6. Versi *HTML*

a. *HTML* 1.0

Ini adalah awal mula dari HTML (pendahulunya). Pada versi ini masih terlihat beberapa kelemahan dan masih sangat sederhana. Kemampuan yang dimiliki oleh versi 1.0 ini hanya terbatas pada heading, paragraph, *hyperText*, list, dan setak tebal atau miring pada teks.

b. *HTML* 2.0

Versi 2.0 pada 14 Januari 1996, pada versi ini ada beberapa tambahan kemampuan diantaranya penambahan form comment, hal ini menyebabkan adanya sebuah interaktif dan mulai dari versi ini yang menjadikan sebuah pioneer dalam perkembangan homepage interaktif.

c. *HTML* 3.0

Dirilis pada 18 Desember 1997 yang sering disebut sebagai HTML+ yang mempunyai kemampuan dalam beberapa fasilitas diantaranya adalah penambahan fitur table dalam paragraph, akan tetapi versi ini tidak bertahan lama.

d. *HTML 3.2*

Dan pada bulan Mei 1996 dikeluarkan versi baru sebagai pengganti dan penyempurnaan versi 3.0 ini yaitu HTML veri 3.2, keluarnya versi ini dikarenakan adanya beberapa kasus yang timbul pada pengembang *Browser* yang telah melakukan pendekatan dengan cara lain yang justru hal tersebut menjadi popular, maka dibakukan versi 3.2 untuk mengakomodasi praktek yang banyak digunakan oleh pengembang *Browser* dan diterima secara umum, dapat dikatakan bahwa versi 3.2 ini merupakan versi 3.0 yang dikembangkan oleh beberapa pengembang *Browser* seperti Netscape dan Microsoft.

e. *HTML 4.0*

Yang terakhir perombakan terjadi pada tahun 1999 tepatnya tanggal 24 Desember yaitu HTML versi 4.0, seperti yang kita kenal HTML pada saat ini penambahan link, meta, imagemaps. Image dan lain-lain sebagai penyempurnaan versi 3.2. Di samping itu versi ini ditambahkan tag-tag baru seperti ABBR, ACRONYM, BUTTON, PARAM, BUTTON, TBODY, THEAD dan lain sebagainya.

f. *HTML 5.0*

Pada tanggal 4 Maret 2010, terdapat sebuah informasi bahwasannya HTML versi 5.0 masih dikembangkan oleh W3C (World Wide Web Consortium) dan IETF (Internet Engineering Task Force) yaitu sebuah organisasi yang menangani HTML sejak versi 2.0.

7. Struktur Dasar **HTML**

Struktur dasar dokumen HTML adalah sebagai berikut :

```
<html>
<head>
<title>Disini Judul Dokumen HTML</title>
</head>
<body>
    Disini penulisan informasi Web
</body>
</html>
```

Dari struktur dasar HTML di atas dapat dijelaskan sebagai berikut:

a. Tag

Adalah teks khusus (*markup*) berupa dua karakter "<" dan ">", sebagai contoh **<body>** adalah tag dengan nama body. Secara umum tag ditulis secara berpasangan, yang terdiri atas **tag pembuka** dan **tag penutup** (ditambahkan karakter "/" setelah karakter "<"), sebagai contoh **<body>** ini adalah tag pembuka isi dokumen HTML, dan **</body>** ini adalah tag penutup isi dokumen HTML.

b. Element

Element terdiri atas tiga bagian, yaitu tag pembuka, isi, dan tag penutup. Sebagai contoh untuk menampilkan judul dokumen HTML pada *web Browser* digunakan element title, dimana: ini adalah tag penutup judul dokumen HTML. Tag-tag yang ditulis secara berpasangan pada suatu element HTML, tidak boleh saling tumpang tindih dengan pasangan tag-tag lainnya.

Contoh penulisan tag-tag yang benar

```
<p>  
<b>  
.....  
</b>  
</p>
```

Contoh penulisan tag-tag yang salah

```
<p>  
<b>  
.....  
</p>  
</b>
```

c. Attribute

Attribute mendefinisikan property dari suatu element HTML, yang terdiri atas nama dan nilai. Penulisannya adalah sebagai berikut:

```
<TAG>  
nama-attr="nilai-attr"  
nama-attr="nilai-attr"  
.....  
>  
.....  
</TAG>
```

Secara umum nilai attribute harus berada dalam tanda petik satu atau dua.

d. Element HTML

Menyatakan pada *Browser* bahwa dokumen *Web* yang digunakan adalah HTML.

Sintaks:

<html>

.....

</html>

e. Element HEAD

Merupakan kepala dari dokumen HTML. Tag dan tag terletak di antara tag dan tag.

Sintaks:

<head>

.....

</head>

f. Element Title

Merupakan judul dari dokumen HTML yang ditampilkan pada judul jendela *Browser*. Tag **<title>** dan tag **</title>** terletak di antara tag **<head>** dan tag **</head>**.

Sintaks:

<title>

.....

</title>

g. Element Body

Element ini untuk menampilkan isi dokumen HTML. Tag <body> dan tag </body> terletak di bawah tag <head> dan tag </head>. Element BODY mempunyai attribute-attribute yang menspesifikasikan khususnya warna dan latarbelakang dokumen yang akan ditampilkan pada *Browser*.

Sintaks:

```
<body Text="v" bgcolor="w" background="uri" link="x"  
alink="y" vlink="z">  
.....  
</body>
```

Attribute *Text* memberikan warna pada teks, bgcolor memberikan warna pada latarbelakang dokumen HTML, background memberikan latar belakang dokumen HTML dalam bentuk gambar, link memberikan nilai warna untuk link, alink memberikan warna untuk link yang sedang aktif, vlink memberikan warna untuk link yang telah dikunjungi. Jika attribute bgcolor dan background keduanya dispesifikasikan maka attribute background yang akan digunakan, akan tetapi jika nilai attribute background (gambar) tidak ditemukan pada dokumen HTML maka attribute bgcolor yang akan digunakan.

8. Kelebihan dan Kekurangan *HTML*

Kelebihan *HTML*:

- a. Merupakan bahasa pengkodean yang lintas platform (cross platform), maksudnya HTML dapat digunakan pada berbagai jenis mesin komputer yang berbeda dan berbagai macam sistem operasi

yang berbeda. Jadi berdifikat fleksibel karena ditulis cukup dengan menggunakan editor karakter ASCII.

- b. Dapat disisipi gambar baik gambar statis atau dinamis (animasi) termasuk menggunakan gambar untuk dijadikan hyperlink. Gambar disini digunakan untuk merujuk pada suatu halaman *web*, dimana setiap titik-titik yang sudah didefinisikan berupa rectangular (kotak), poligon (kurva tak beraturan) atau lingkaran digunakan untuk ‘jump’ ke halaman lain, atau link ke halaman di luar *web* yang bersangkutan.
- c. Dapat disisipi animasi berupa Java Applet atau *File-File* animasi dari Macromedia Flash atau Macromedia Shockwave (untuk keperluan ini, *Browser* harus memiliki plug-in khusus untuk menjalankan *File-File* animasi ini).
- d. Dapat disisipi bahasa pemrograman untuk mempercantik halaman *web* seperti Javascript, Vbscript, Active Server Pages, Perl, Tcl, *PHP*, dan sebagainya.
- e. Bukan merupakan bahasa pemrograman jadi tidak memerlukan kompiler. Cara menjalankannya cukup dengan menggunakan *Browser*.

Kekurangan *HTML*:

- a. Menghasilkan halaman yang statis, untuk memperoleh halaman yang dinamis harus menggunakan bahasa pemrograman tertentu seperti Javascript atau Vbscript dan animasi seperti Flash atau Shockwave.
- b. Memiliki tag-tag yang begitu banyak sehingga susah dipelajari untuk yang masih awam.
- c. Tidak dapat menghasilkan halaman yang interaktif. Interaktif disini maksudnya *Client* dapat berinteraksi dengan *Server*. Untuk keperluan itu HTML harus disisipi bahasa pemrograman yang dapat menangani hal tersebut, contohnya Perl dan Tcl.

3.2.5 Hubungan *HTML* dan *PHP*

Halaman *web* biasanya disusun dari kode-kode html yang disimpan dalam sebuah *File* berekstensi .html. *File* html ini dikirimkan oleh *Server* (atau *File*) 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. Program ini dapat berdiri sendiri ataupun disisipkan di antara kode-kode html sehingga dapat langsung ditampilkan bersama dengan kode-kode html tersebut. Program *PHP* dapat ditambahkan dengan mengapit program tersebut diantara tanda.

Tanda-tanda tersebut biasanya disebut tanda untuk escaping (kabur) dari kode html. *File* html yang telah dibubuh program *PHP* harus diganti ekstensi-nya menjadi *.PHP3* atau *PHP*. *PHP* merupakan bahasa pemograman *web* yang bersifat *Server-side* HTML=embedded scripting, di mana script-nya menyatu dengan HTML dan berada si *Server*. Artinya adalah sintaks dan perintah-perintah yang kita berikan akan sepenuhnya dijalankan di *Server* tetapi disertakan HTML biasa. *PHP* dikenal sebagai bahasa scripting yang menyatu dengan tag HTML, dieksekusi di *Server* dan digunakan untuk membuat halaman *web* yang dinamis seperti ASP (Active *Server* Pages) dan JSP (Java *Server* Pages).

BAB IV

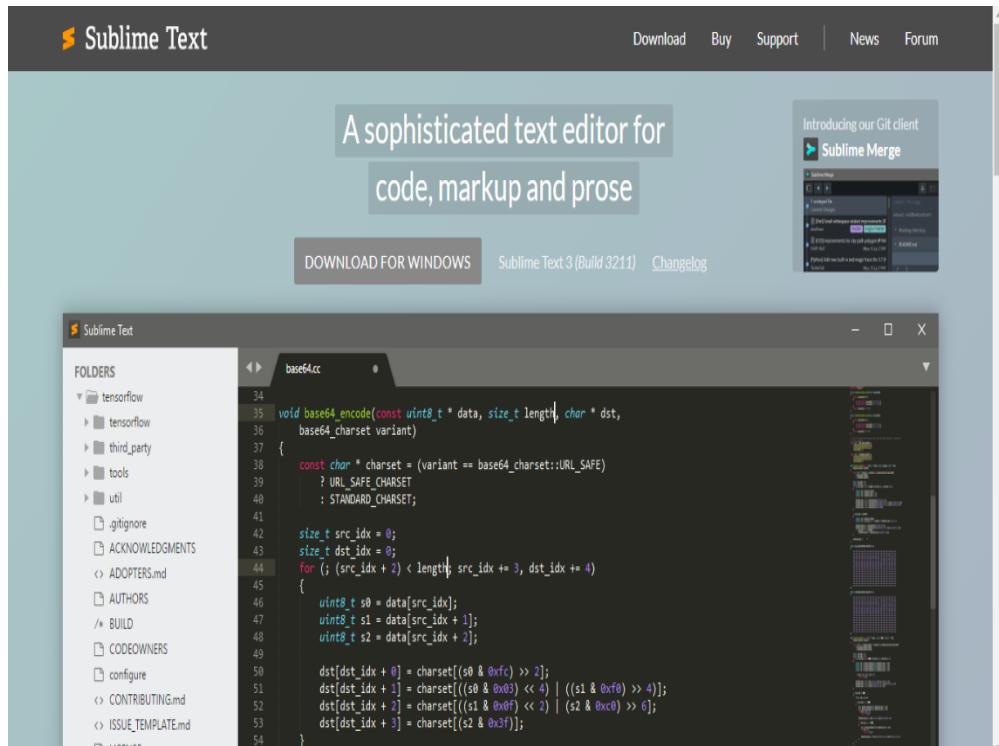
INSTALASI TOOLS YANG DIGUNAKAN

4.1 Instalasi *Tools* Yang Digunakan

4.1.1 *Sublime Text*

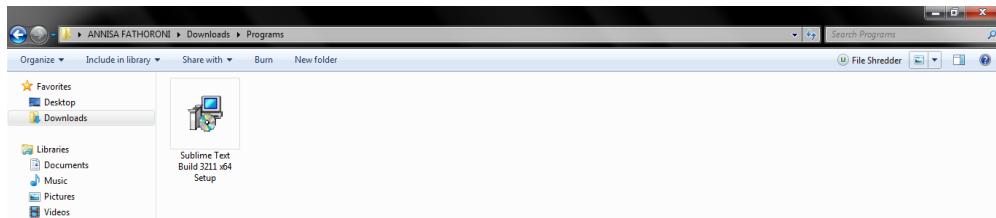
Berikut ini merupakan tahapan-tahapan yang dilakukan untuk instalasi *Sublime Text*, yaitu:

1. Tahapan yang pertama untuk melakukan instalasi maka download dahulu *File .exe* pada <https://www.SublimeText.com/>.



Gambar 4. 1 Website Sublime

Tahapan yang kedua setelah selesai mendownload maka buka *Directory* dimana *File .exe* tersimpan, kemudian Double klik *File .exe* hasil download:



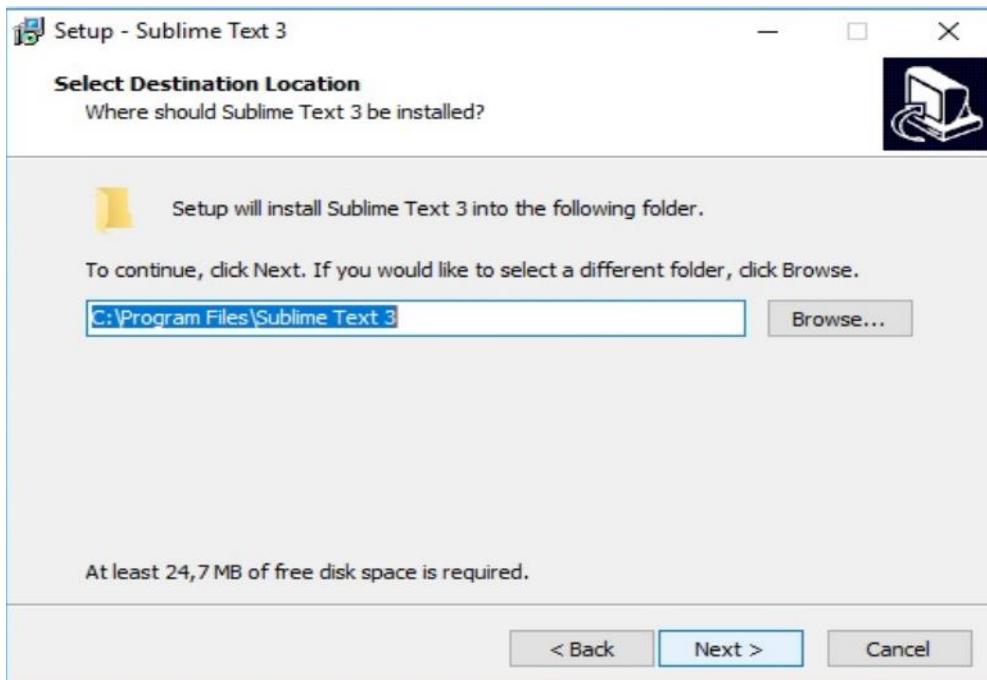
Gambar 4. 2 Letak *Directory File .exe* Tersimpan

Tahapan ketiga, jika sudah double klik *File .exe* maka akan muncul tampilan berikut, kemudian klik next:



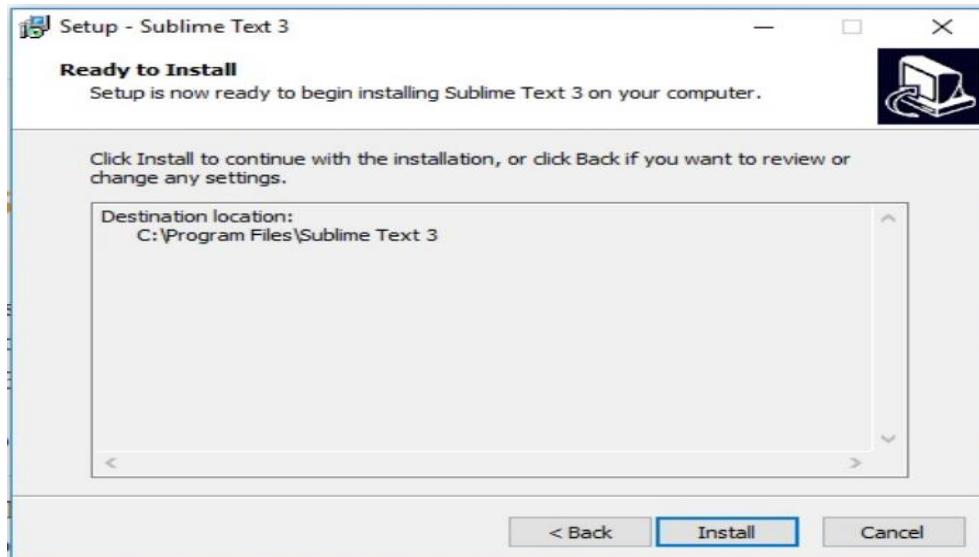
Gambar 4. 3 Proses Instalasi *Sublime*

2. Selanjutnya, klik next:



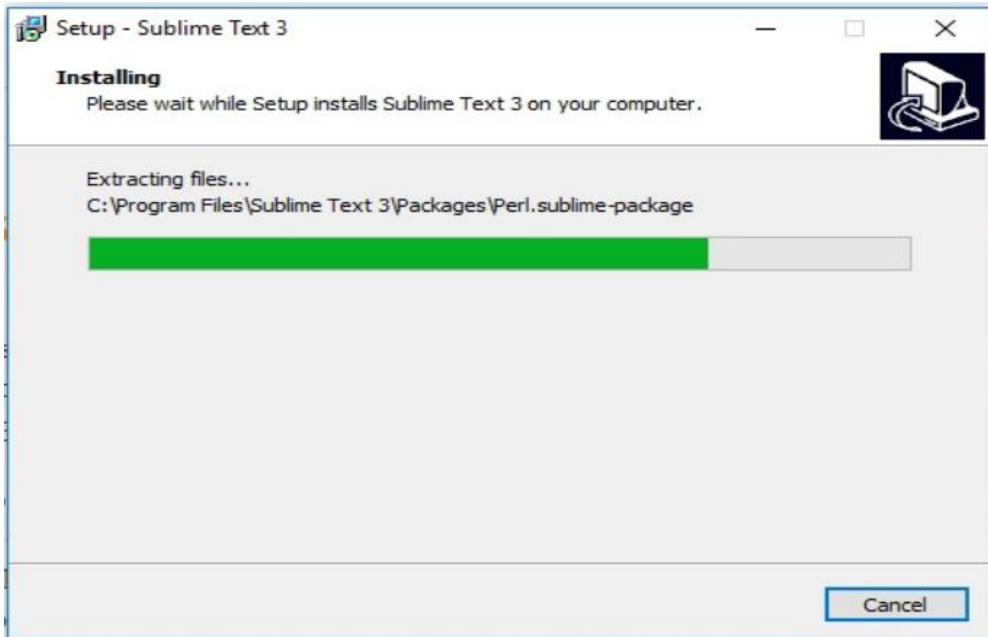
Gambar 4. 4 Proses Instalasi *Sublime*

3. Selanjutnya, klik next:



Gambar 4. 5 Proses Instalasi *Sublime*

4. Tunggu sampai proses instalasi selesai, Setelah proses ini selesai maka *Sublime* telah bisa digunakan



Gambar 4. 6 Proses Instalasi *Sublime*

4.1.2 XAMPP

Berikut ini merupakan tahapan-tahapan yang dilakukan untuk instalasi *XAMPP*, yaitu:

1. *Download Package XAMPP* pada situ resminya,

The screenshot shows the Apache Friends website with a blue header bar containing links for 'Apache Friends', 'Download' (which is highlighted), 'Add-ons', 'Hosting', 'Community', and 'About'. There is also a search bar and a language selection for 'EN'. Below the header, a large 'Download' button is prominently displayed. To its left, there is a note: 'XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.' On the right, there is a 'Documentation/FAQs' section with a note about no real manual and links to Linux, Windows, OS X, and OS X XAMPP-VM FAQs. Below the main content area, there is a 'Add-ons and Themes' section with icons for various add-ons.

XAMPP for Windows 7.2.26, 7.3.13 & 7.4.1

Version	Checksum	Size
7.2.26 / PHP 7.2.26	What's Included? md5 sha1	Download (64 bit) 145 Mb
7.3.13 / PHP 7.3.13	What's Included? md5 sha1	Download (64 bit) 146 Mb
7.4.1 / PHP 7.4.1	What's Included? md5 sha1	Download (64 bit) 146 Mb

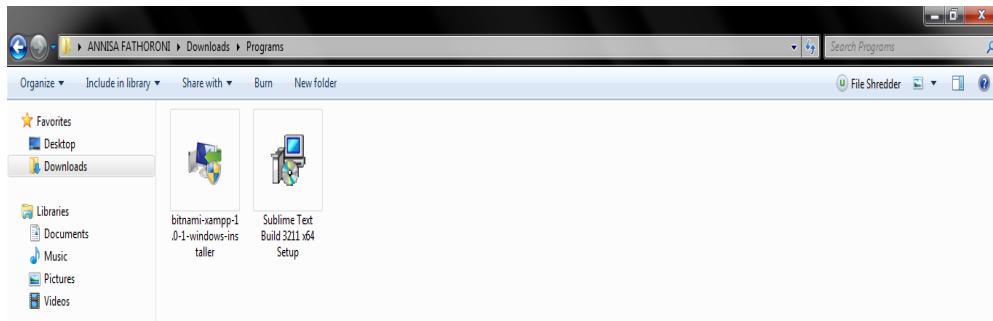
Requirements Add-ons More Downloads »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these

<https://www.apachefriends.org/download.html>

Gambar 4. 7 Website XAMPP

2. Tahapan yang kedua setelah selesai mendownload maka buka *Directory* dimana *File .exe* tersimpan, kemudian Double klik *File .exe* hasil download:



Gambar 4. 8 Letak Directory

3. Lalu selanjutnya akan muncul *splash screen bitnami*, setelah itu akan ada peringatan *error*, ini terjadi terkait masalah *permission* atau hak akses, klik OK karena nantinya akan di konfigurasi ke *hard drive* lain.



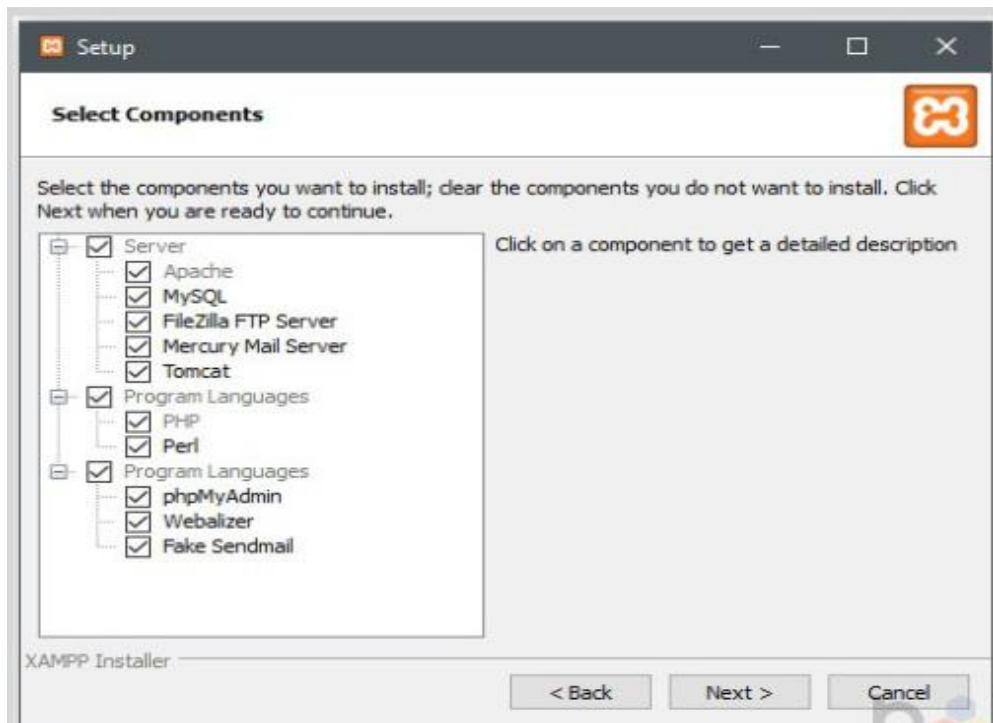
Gambar 4. 9 Proses Instalasi XAMPP

4. Setelah itu akan muncul bagian awal *instalasi Setup Wizard xamp*, klik *Next* untuk memulai instalasi



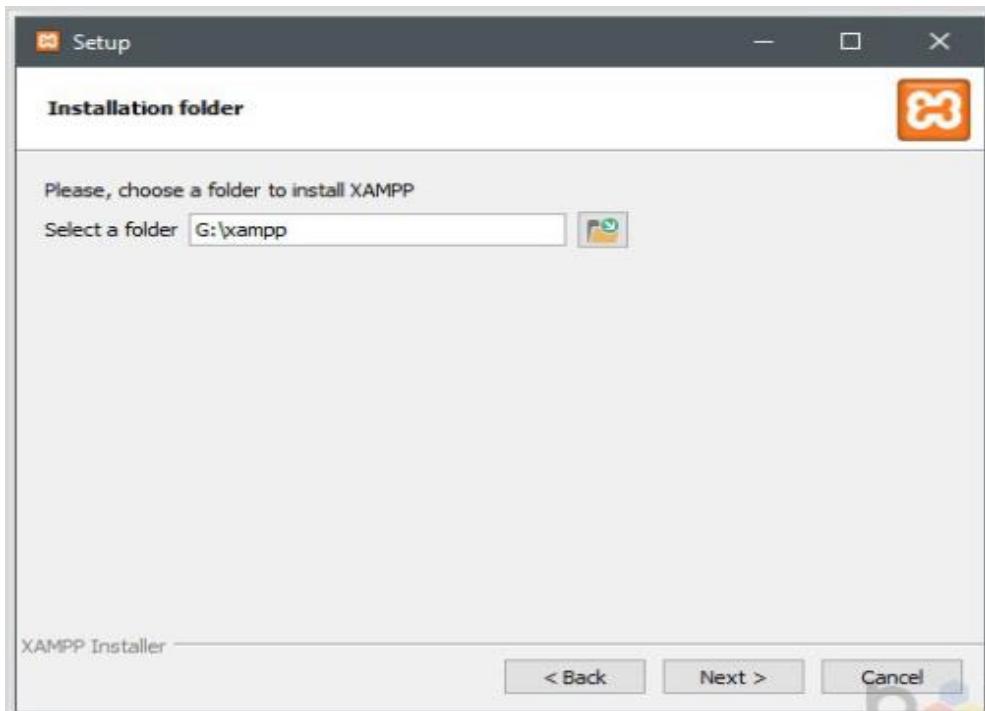
Gambar 4. 10 Proses *Instalasi XAMPP*

5. Setelah itu akan muncul halaman pemilihan komponen apa saja yang ingin di *install* di dalam *web Server XAMPP* ini, silahkan untuk di sesuaikan, anda dapat meminimalisir modul-modul instalasi sesuai kebutuhan, tetapi di dalam tutorial ini saya membiarkan semuanya terinstall. jika sudah klik *Next*.



Gambar 4. 11 Proses Instalasi XAMPP

6. Selanjutnya akan di arahkan untuk penempatan folder instalasi *XAMPP*, jika secara bawaan akan di arahkan ke drive C, karena masalah hak akses yang terjadi di awal tadi sehingga perlu di arahkan ke hard drive lain, setelah itu klik Next.



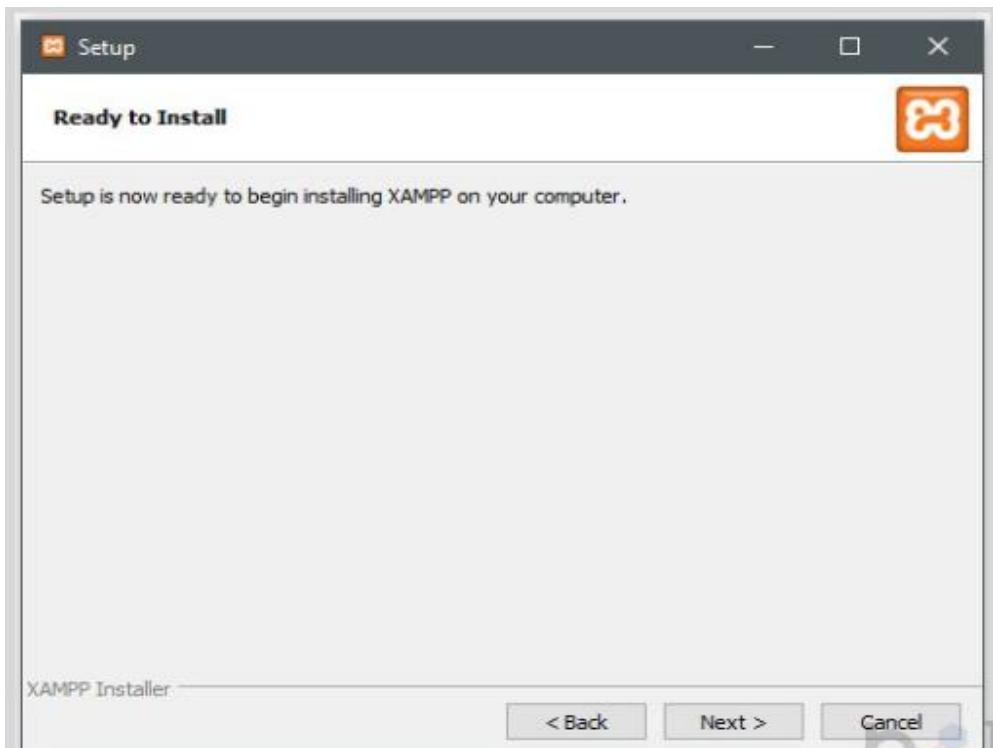
Gambar 4. 12 Proses Instalasi *XAMPP*

7. Langkah berikutnya adalah tampilan promosi Bitnami menawarkan cara menginstall CMS seperti *WordPress* dan lain-lain. Pada langkah ini ada sebuah *checkbox* pada bagian *Learn more about Bitnami for XAMPP* silahkan di hilangkan ceklis yang secara bawaan telah ada jika tidak ingin mengikuti petunjuk instalasi CMS. jika sudah klik *Next*.



Gambar 4. 13 Proses Instalasi XAMPP

8. Pada langkah berikutnya *Ready to Install* ini sebenarnya hanya untuk memastikan apakah anda sudah yakin pada tahap-tahap sebelumnya, jika di rasa anda kurang yakin bisa menekan tombol back, tetapi dalam tutorial ini telah di uji coba langkah-langkah instalasi *XAMPP* sebagai *web Server* lokal di tutorial ini. Jika sudah klik *Next*.



Gambar 4. 14 Proses Instalasi *XAMPP*

9. Selanjutnya adalah proses *extract File-File XAMPP* ke dalam *hard drive folder* yang telah di tetapkan sebelumnya, tunggu sampai proses instalasi selesai dimana proses ini tidak memakan waktu lama. Jika sudah klik *Next*.



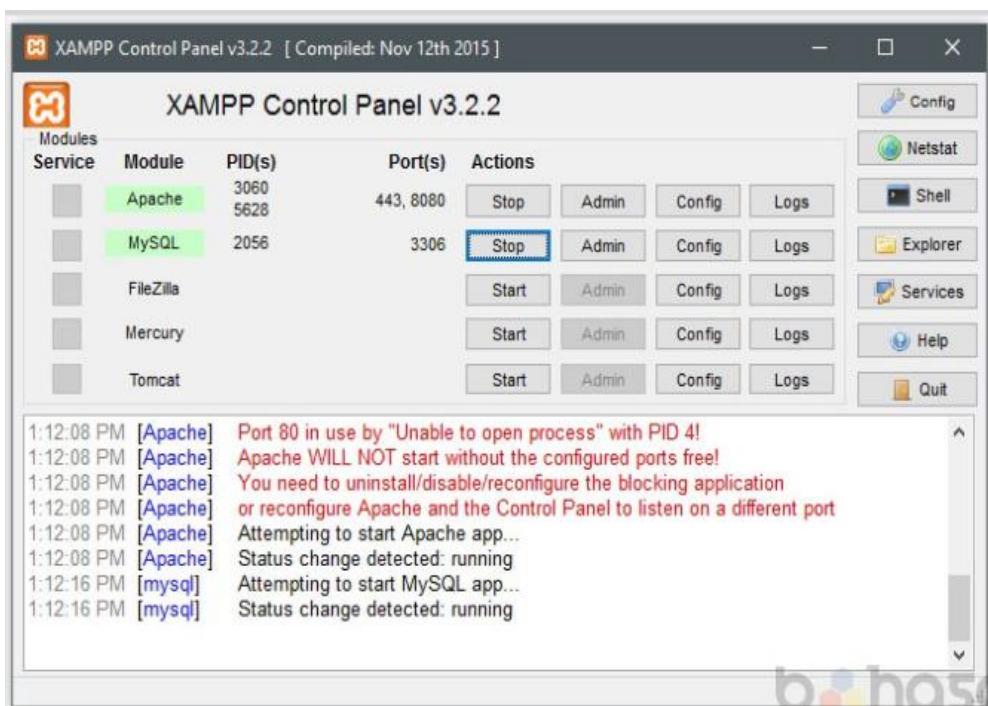
Gambar 4. 15 Proses Instalasi XAMPP

10. Langkah berikutnya anda akan di bawa ke bagian *Completing the XAMPP Setup Wizard* ini adalah langkah terakhir dalam rangkaian proses instalasi. Terdapat *checkbox* *Do you want to start the Control Panel now?* biarkan saja ceklis tersebut, atau jika tidak ada ceklis maka silahkan berikan ceklis agar setelah ini *XAMPP Control Panel* secara otomatis berjalan. Jika sudah klik *finish*.



Gambar 4. 16 Proses Instalasi XAMPP

11. Sampai disini proses instalasi telah selesai, bisa mengecek langsung ke folder yang sebelumnya telah di arahkan. Karena pada tahap terakhir kita memberikan ceklis agar *XAMPP Control Panel* secara otomatis tampil maka tunggu sampai *Control Panel* tersebut muncul. Jika *Control Panel XAMPP* sudah muncul silahkan klik *start* pada *Apache* dan *Mysql* hingga tulisan *Apache* dan *Mysql* di *block* warna hijau.



Gambar 4. 17 Tampilan XAMPP

12. Untuk menguji jika semua sudah berjalan dengan baik silahkan buka *Browser* dan arahkan ke alamat <http://localhost/> atau <http://127.0.0.1/> ini adalah alamat umum komputer lokal. Jika berhasil seharusnya diarahkan ke halaman *XAMPP*.

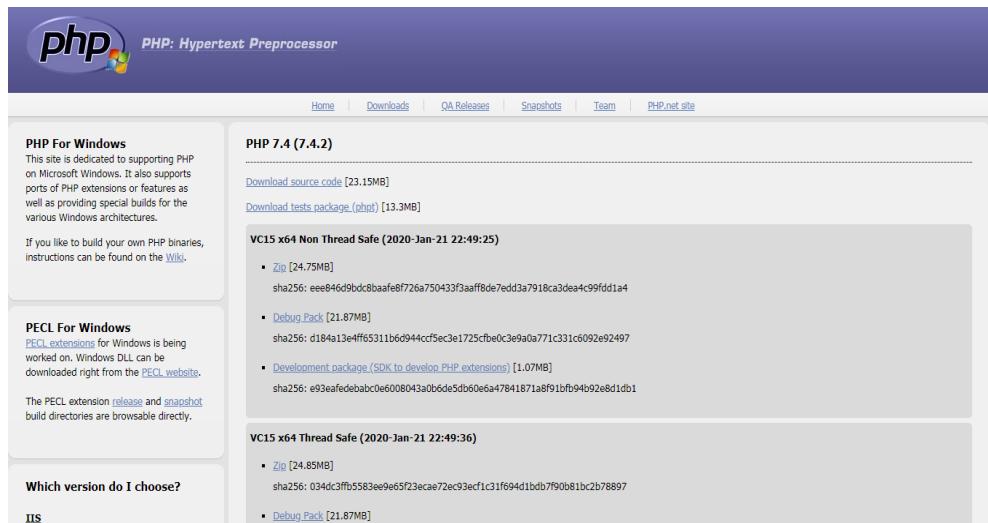


Gambar 4. 18 Hasil Akhir *XAMPP*

4.1.3 PHP

Berikut ini merupakan tahapan-tahapan yang dilakukan untuk instalasi *PHP*, yaitu:

1. Berikut ini merupakan tahapan-tahapan yang dilakukan untuk instalasi *PHP*, yaitu:



Gambar 4. 19 Website PHP

2. Buat *folder* baru di "C:\Apache" namainya dengan "*PHP*" jadi *folder pathnya* seperti berikut "C:\Apache\PHP". *Copy semua File dan folder* yang telah di *unzip* ke "C:\Apache\PHP". Dalam *folder* "C:\Apache\PHP" cari dan *rename* "PHP.ini-production" ke "PHP.ini".

Double click PHP.ini untuk membuka dan cari baris berikut:

```
doc_root =
```

Tambahkan *path* ke *Document Root*, bisa di cari pada *Apache konfigurasi*:

```
doc_root = "D:\My Websites"
```

Kemudian cari baris berikut:

```
; extension_dir = "ext"
```

Buang komen dan tambahkan *path* ke dalam *direktori extensi PHP*:

```
extension_dir = "C:\Apache\PHP\ext"
```

3. Secara *default PHP* akan menggunakan windows *temp* sebagai *folder temporer*. Bila terdapat *error* dalam *script* secara kebetulan aku mengosongkan *System foldet temp*. Jadi mungkin ini adalah solusi yang bagus untuk membuat *folder temp* yang terpisah untuk *PHP*. Buat *folder* baru di "C:\Apache" namai foldernya dengan "temp" didalam "temp" buat 2 folder degan nama "upload" dan "session" Folder baru yakni "C:\Apache\temp\upload" dan "C:\Apache\temp\session"

Dalam *File konfigurasi PHP (PHP.ini)* cari baris berikut:

```
;upload_tmp_dir =
```

Buang komenya dan tambahkan *path* untuk *folder upload*:

```
upload_tmp_dir= "C:\Apache\temp\upload"
```

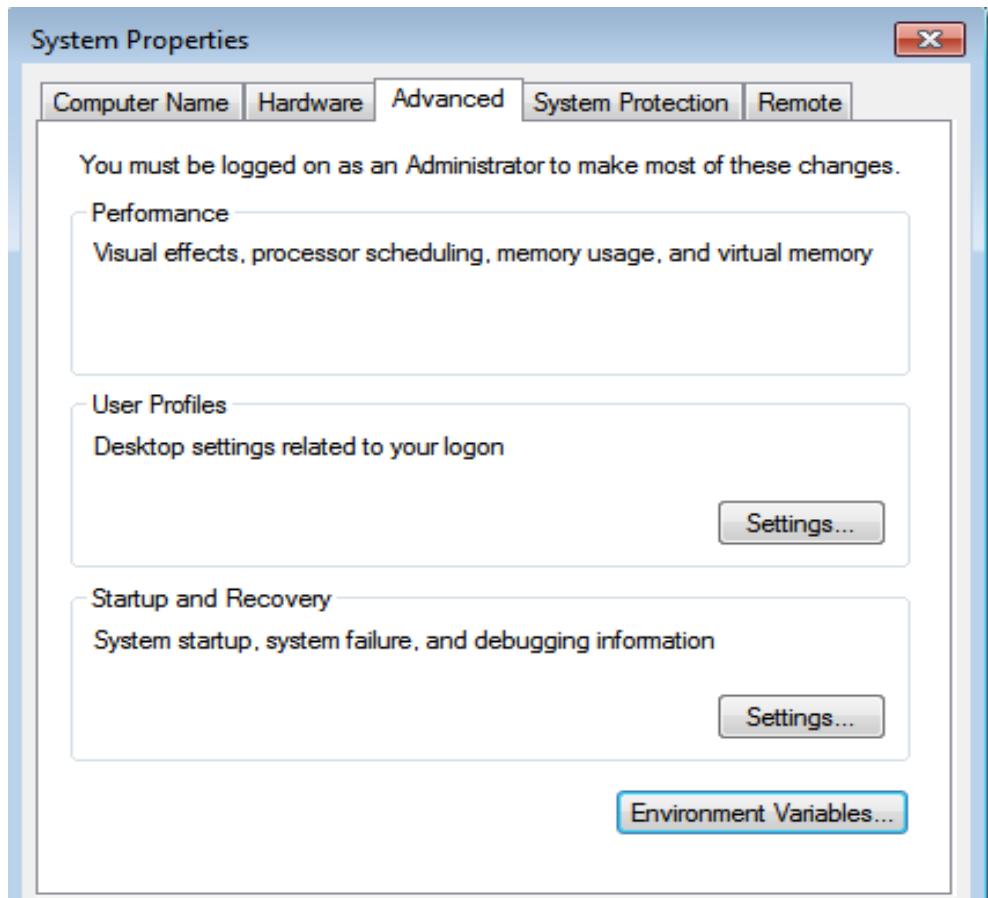
Dan kemudian baris berikut:

```
;session.save_path = "/tmp"
```

Buang komennya dan tambahkna *path* untuk *folder session*:

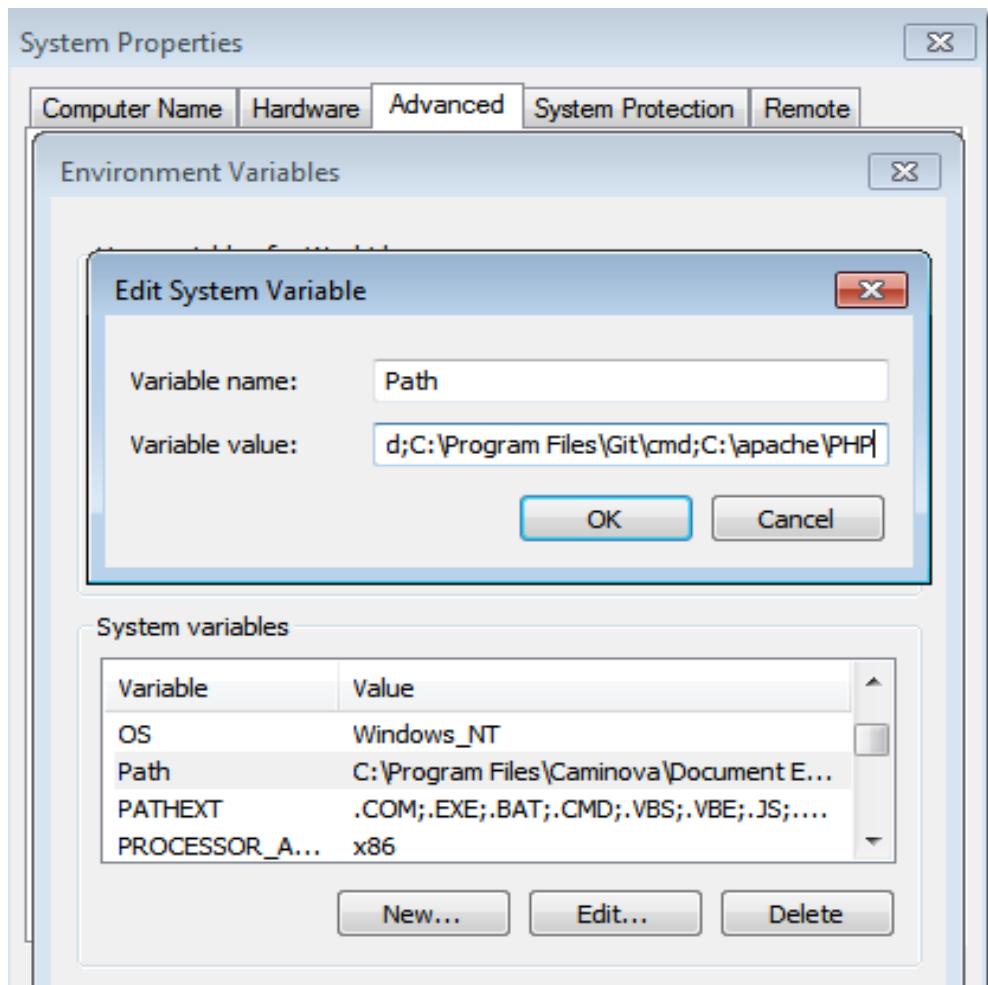
```
session.save_path= "C:\Apache\temp\session"
```

4. Menambahkan *PHP* ke *path System*. Buka *System Properties*, untuk membukanya bisa dengan cara klik kanan pada *My Computer* dan klik *Properties*, pada *Home Menu Control Panel* klik *Advanced System Settings*. Pada tab *Advanced* Klik *Environment Variables*.



Gambar 4. 20 *System Properties*

5. Dalam "System Variable" double klik "Path" Edit variabel dan tambahkan path "C:\Apache\PHP" kedalam nilai variabel, click OK untuk menyimpan konfigurasi.



Gambar 4. 21 *Edit System Variabel*

6. Komputer harus *restart* untuk melihat *effect*-nya. Menghubungkan *Apache Server* dengan *PHP*. Untuk menghubungkan *Apache Server* dengan *PHP* kita harus meng*Edit Apache Configuration* yang terletak di C:\Apache\conf\httpd.conf

Cari *Directory Index* seperti berikut:

```
<IfModule dir_module> DirectoryIndex index.html  
</IfModule>
```

Tambahkan *index.PHP* seperti berikut ini

```
<IfModule dir_module> DirectoryIndex index.html  
index.php </IfModule>
```

```
237 # DirectoryIndex: sets the file that Apache will serve if  
238 # is requested.  
239 #  
240 <IfModule dir_module>  
241     DirectoryIndex index.html index.php  
242 </IfModule>  
243 #  
244 #
```

Gambar 4. 22 Menambahkan *Source Code*

Tambahkan baris berikut diantara *<IfModule mime_module>* dan *</IfModule>*

```

ScriptAlias      /PHP/      "C:/Apache/PHP/"      AddType
application/x-httdp-PHP      .PHP      .PHP5      Action
application/x-httdp-PHP      "/PHP/PHP-cgi.exe"      SetEnv
PHPRC "C:/Apache/PHP"

382      AddType application/x-compress .Z
383      AddType application/x-gzip .gz .tgz
384
385      ScriptAlias /php/ "C:/Apache/php/"
386      AddType application/x-httdp-php .php .php5
387      Action application/x-httdp-php "/php/php-cgi.exe"
388      SetEnv PHPRC "C:/Apache/php"
389
390      #

```

Gambar 4. 23 Menambahkan *Source Code*

Tambahkan baris berikut pada bagian akhir httpd.conf

```

PHPIniDir      "C:/Apache/PHP/"      LoadModule      PHP5_module
"C:/Apache/PHP/PHP5apache2_2.dll"

493 <IfModule ssl_module>
494 SSLRandomSeed startup builtin
495 SSLRandomSeed connect builtin
496 </IfModule>
497
498 PHPIniDir "C:/Apache/php/"
499 LoadModule php5_module "C:/Apache/php/php5apache2_2.dll"

```

Gambar 4. 24 Menambahkan *Source Code*

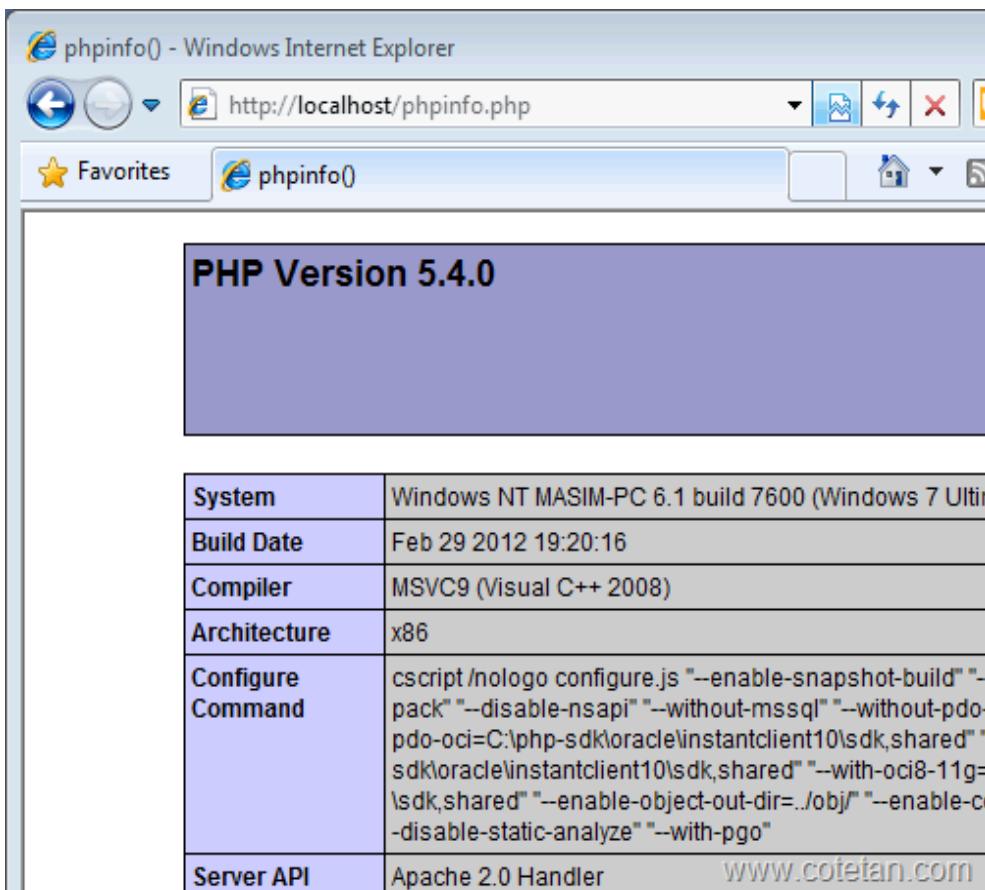
Jika menggunakan Apache v2.4, ubahlah baris terakhir *module* yang unggah "PHP5apache2_4.dll" .

```
LoadModule PHP5_module  
"C:/Apache/PHP/PHP5apache2_4.dll"
```

7. *Restart Apache Server, test PHP.* Buka Notepad, copy paste kode dibawah ini:

```
<?PHP PHPinfo(); ?>
```

8. Simpan File kedalam root dokumen Apache "D:\My Websites" as "PHPinfo.PHP" Buka Browser, pada address bar ketikan "http://localhost/PHPinfo.PHP" dan tekan Enter.



Gambar 4. 25 Test PHP

BAB V

PERANCANGAN PEMBUATAN APLIKASI

Sebelum melakukan tutorial penulis akan menjelaskan sedikit pemaparan latar belakang mengapa SISTEM *MONITORING TERHADAP JOB DESK OPERATIONAL HUMAN* dibuat dan hasil yang diperoleh dari sistem yang sudah berhasil dibuat dan diuji. Model pengembangan yang digunakan pada penelitian ini ialah model *Waterfall*. Model *Waterfall* merupakan model pengembangan sistem informasi yang sistematik dan sekuensial. Alasan penggunaan model *Waterfall* sebagai metode pengembangan sistem informasi ialah kebutuhan terdefinisi secara jelas dan tahap-tahap pada model *Waterfall* terstruktur secara jelas.

5.1 Analisis

Analisis terhadap proses bisnis Sistem *Monitoring evaluasi kantor pusat ke kantor wilayah* yang selanjutnya dinamakan Sistem *Monitoring Job Desk Human Capital* yang dimulai dengan pemberian tugas hingga penyelesaian tugas yang diberikan.

5.1.1 Analisis Sistem yang Sedang Berjalan

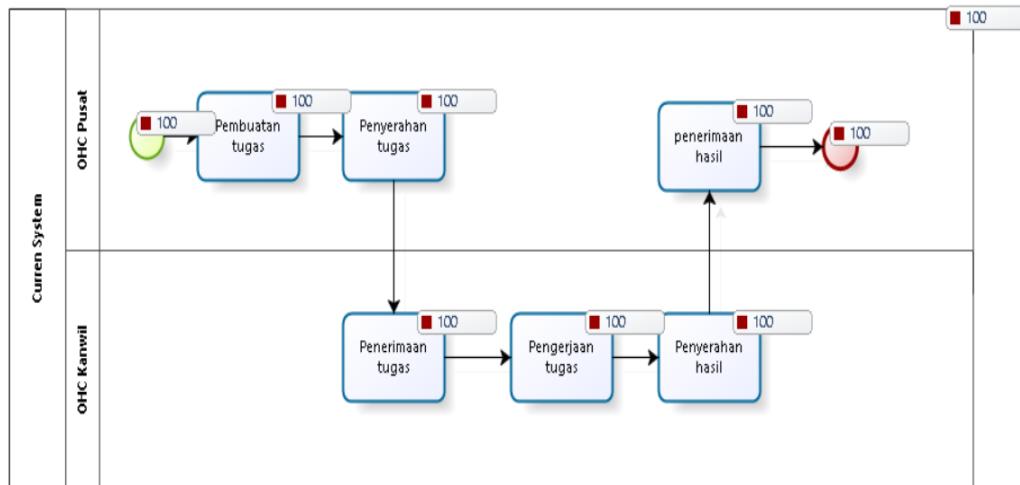
Dari hasil wawancara dan diskusi yang telah dilakukan maka alur yang sedang berjalan pada Sistem *Monitoring evaluasi kantor pusat ke kantor wilayah* adalah sebagai berikut:

5.1.1.1 Analisis Prosedur Sistem Berjalan

Assitant Manager II di OHC kantor pusat membuat tugas, kemudian Assitant Manager II memberikan tugas ke Staff OHC kantor wilayah. Setelah itu, Staff OHC kantor wilayah menerima tugas lalu mengerjakan tugas tersebut. Setelah tugas yang diberikan selesai, maka Staff OHC kantor wilayah

mengirim kembali hasil akhir dari tugas yang diberikan ke Assitant Manager II di OHC kantor pusat, proses selesai.

Adapun *Bizagi modeler* sistem berjalan pada Sistem *Monitoring evaluasi* kantor pusat ke kantor wiliyah adalah sebagai berikut:



Gambar 5. 1 *Bizagi modeler sistem berjalan pada Sistem Monitoring evaluasi kantor pusat ke kantor wiliyah*

Tabel 5. 1 Hasil simulasi proses sistem berjalan pada Sistem *Monitoring evaluasi* kantor pusat ke kantor wiliyah

Name	Type	Instances completed
Curren System	Process	100
NoneStart	Start event	100
Pembuatan tugas	Task	100
Penyerahan tugas	Task	100
Penerimaan tugas	Task	100
Pengerjaan tugas	Task	100
Penyerahan hasil	Task	100
penerimaan hasil	Task	100
NoneEnd	End event	100

5.1.1.2 Analisis Dokumen yang Digunakan

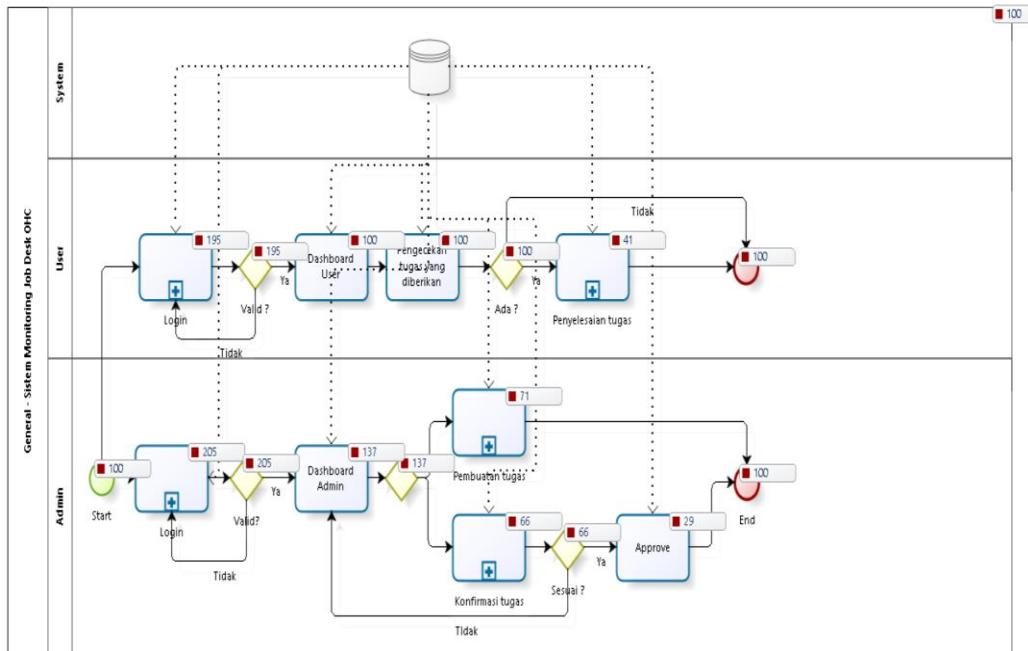
Dari hasil analisis yang dilakukan, dokumen yang digunakan oleh Sistem *Monitoring* evaluasi kantor pusat ke kantor wiliyah diantaranya dokumen Monev Kantor wilayah. Adapun dokumen yang dimaksud adalah sebagai berikut:

Tabel 5. 2 Dokumen Monev Kantor Wilayah

Dibuat oleh	Assitant Manager II
Dibuat untuk	Staff OHC kantor wilayah
Isi	Berupa tugas yang diberikan OHC pusat.
Frekuensi	Dibuat sesuai tugas yang ada.
Tujuan	Menyelesaikan tugas yang diberikan.

5.1.2 Analisis Sistem yang akan Dibangun

Sub proses terdiri atas sub proses *Login*, sub proses pembuatan tugas, sub proses konfirmasi tugas, sub proses penyelesaian tugas. Adapun *Bizagi modeler* yang akan dibangun adalah sebagai berikut:



Gambar 5. 2 *Bizagi modeler sistem yang akan dibangun pada Sistem Monitoring Job Desk OHC.*

Tabel 5. 3 Hasil simulasi proses sistem yang akan dibangun pada Sistem Monitoring Job Desk OHC.

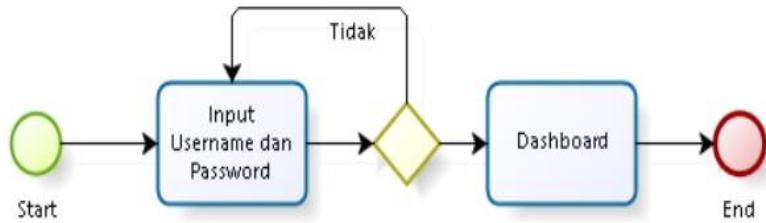
Name	Type	Instances completed
General - Sistem Monitoring Job Desk OHC	Process	100
Start	Start event	100
Valid?	Gateway	205
Dashboard Admin	Task	137
End	End event	100
Valid ?	Gateway	195
Dashboard Users	Task	100
Ada ?	Gateway	100
NoneEnd	End event	100
ExclusiveGateway	Gateway	137

Sesuai ?	Gateway	66
Approve	Task	29
Pengecekan tugas yang diberikan	Task	100
<i>Login</i>	Process	195
<i>Login</i>	Process	205
Pembuatan tugas	Process	71
Penyelesaian tugas	Process	41
Konfirmasi tugas	Process	66

1. Analisis Sistem Yang Akan Dibangun Pada Sub Proses *Login*

Pengguna melakukan proses *Login* untuk masuk ke dalam sistem.

Adapun *Bizagi modeler* pemesanan barang sebagai berikut :



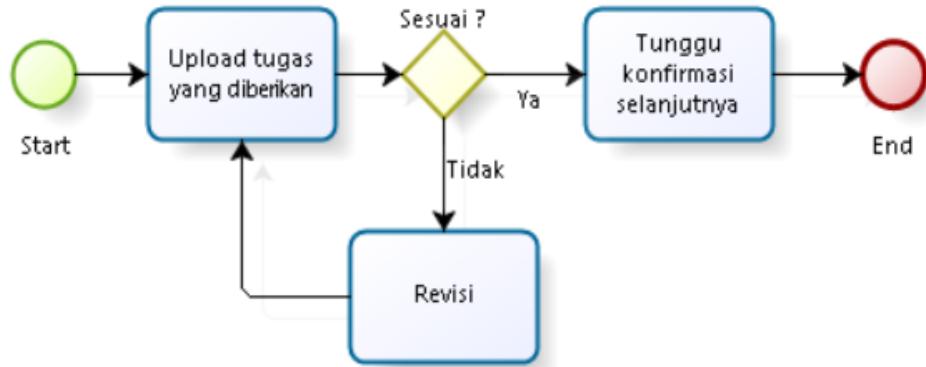
Gambar 5. 3 *Bizagi modeler* sub proses *Login* sistem yang akan dibangun.

Tabel 5. 4 *Hasil simulasi sub proses Login sistem yang akan dibangun.*

Name	Type	Instances completed
<i>Login</i>	Process	100
Start	Start event	100
End	End event	100
<i>Input Username dan Password</i>	Task	298
<i>Valid?</i>	Gateway	298
<i>Login Berhasil</i>	Task	100
Registrasi	Task	92

2. Analisis Sistem Yang Akan Dibangun Pada Sub Penyelesaian Tugas

Users melakukan penyelesaian tugas di *system*. Adapun *Bizagi modeler* sub proses penyelesaian tugas sebagai berikut:



Gambar 5. 4 *Bizagi modeler* sub proses penyelesaian tugas.

Tabel 5. 5 Tabel simulasi sub proses penyelesaian tugas yang akan dibangun

Name	Type	Instances completed
Penyelesaian tugas	Process	41
Start	Start event	41
Upload tugas yang diberikan	Task	74
Sesuai ?	Gateway	74
Revisi	Task	33
Tunggu konfirmasi selanjutnya	Task	41
End	End event	41

3. Analisis Sistem Yang Akan Dibangun Pada Sub Pembuatan Tugas

Admin masuk ke dalam *System* (sudah *Login*) kemudia memilih megisi data tugas yang diberikan kepada Staff OHC kantor wilayah. Setelah mengisi data tugas yang diberikan *Admin* mengupload File tugas yang akan di berikan kepada Staff OHC kantor wilayah. Adapun *Bizagi modeler* sub proses pembuatan tugas adalah sebagai berikut:



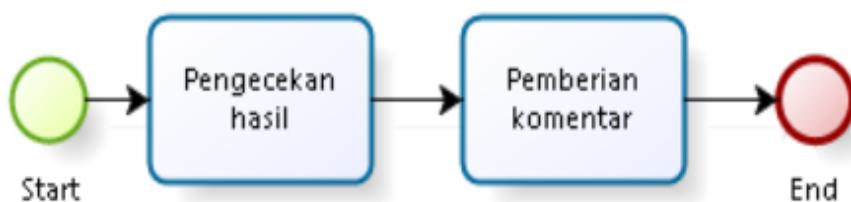
Gambar 5. 5 *Bizagi modeler* sub proses pembuatan tugas yang akan dibangun.

Tabel 5. 6 Tabel simulasi sub proses pembuatan tugas yang akan dibangun

Name	Type	Instances completed
Pembuatan tugas	Process	71
Start	Start event	71
Mengisi data tugas yang diberikan	Task	71
Upload tugas	Task	71
End	End event	71

4. Analisis Sistem Yang Akan Dibangun Pada Sub Proses Konfirmasi Tugas

Admin (Administrator) masuk ke dalam *System* (sudah *Login*), melakukan proses pengecekan hasil dari tugas yang sudah diberikan dan memberikan pada tugas tersebut. Adapun *Bizagi modeler* dari sub proses konfirmasi tugas ini adalah sebagai berikut:



Gambar 5. 6 *Bizagi modeler* sub proses konfirmasi tugas yang akan dibangun.

Tabel 5. 7 Tabel simulasi sub proses Konfirmasi tugas yang akan dibangun.

Name	Type	Instances completed
Konfirmasi tugas	Process	66
Start	Start event	66
Pengecekan hasil	Task	66
Pemberian komentar	Task	66
End	End event	66

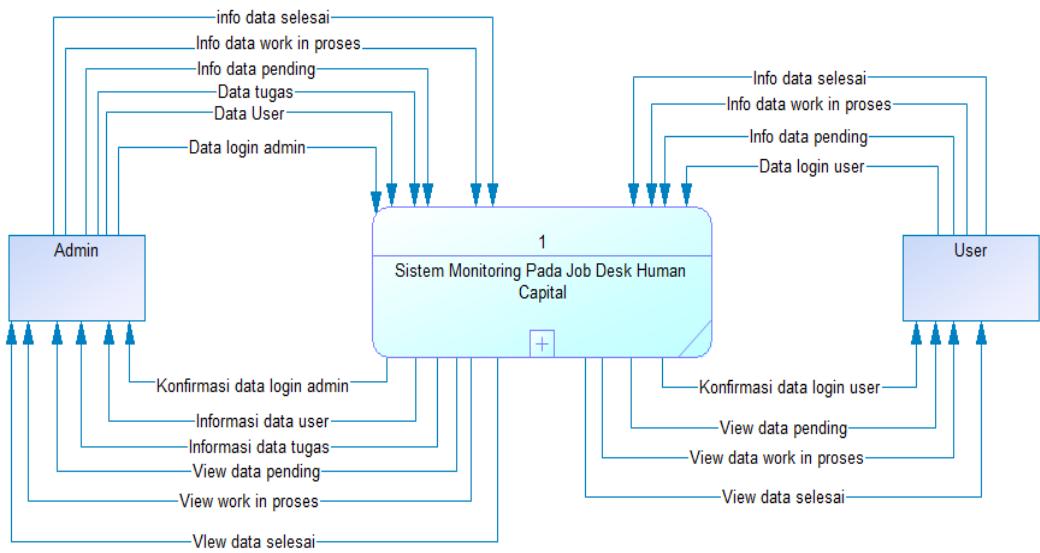
5.1.3 Kondisi Perangkat Keras (hardware) dan Perangkat Lunak (software) yang Berjalan

Kondisi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang berjalan merupakan kondisi dimana pada saat pembangunan aplikasi ini penulis menggunakan perangkat keras dan perangkat lunak saat itu.

5.2 Perancangan

5.2.1 ConText Diagram atau DFD Level 1

ConText diagram adalah sebuah *diagram* sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem. *ConText diagram* direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. Adapun rancangan *ConText Diagram* Sistem *Monitoring Job Desk OHC* dapat dilihat pada gambar berikut:



Gambar 5. 7 *ConText Diagram* Sistem Monitoring Job Desk OHC

Tabel 5. 8 Data in, Data out *conText diagram*

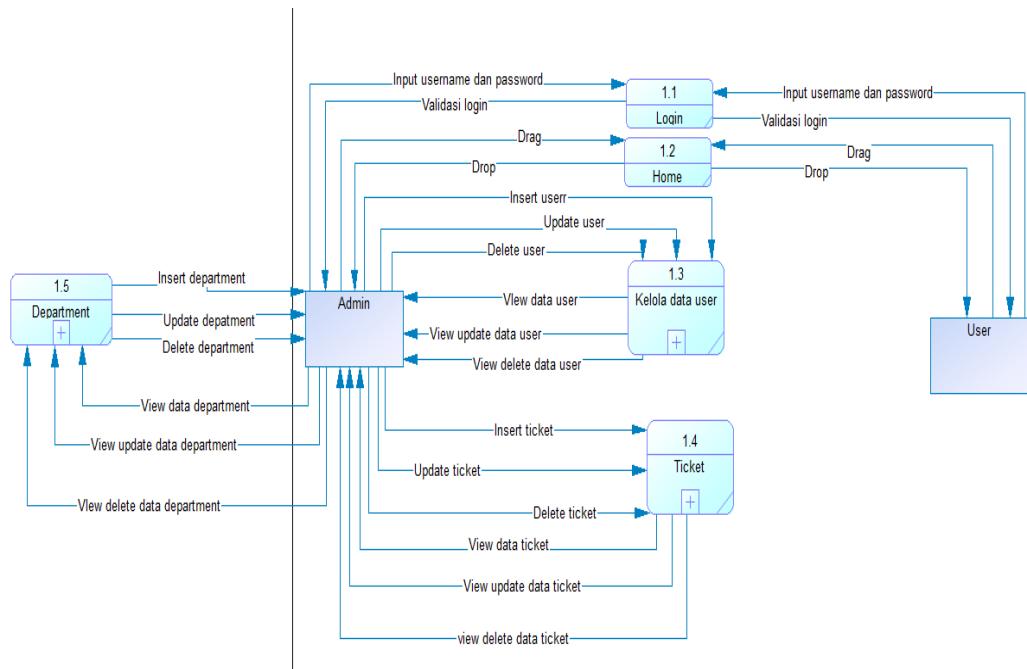
Proses	Data in	Data I
<i>Sistem Monitoring Job Desk Human Capital</i>	<i>Data Login Admin</i>	<i>Konfirmasi Data Login Admin</i>
	<i>Data Users</i>	<i>Informasi Data Users</i>
	<i>Data Tugas</i>	<i>Informasi Data Tugas</i>
	<i>Info data pending</i>	<i>View data pending</i>
	<i>Info data work in proses</i>	<i>View Work in Proses</i>
	<i>Info data selesai</i>	<i>View data selesai</i>
	<i>Data Login Users</i>	<i>Konfirmasi data Login anggota</i>

	Info data pending	<i>View data pending</i>
	Info data work in proses	<i>View work in proses</i>

5.2.2 Data Flow Diagram (DFD)

Sistem yang akan dibangun dalam aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital* dapat digambarkan melalui Data Flow Diagram (DFD) sebagai berikut :

5.2.2.1 Data Flow Diagram Level 2 Aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital*

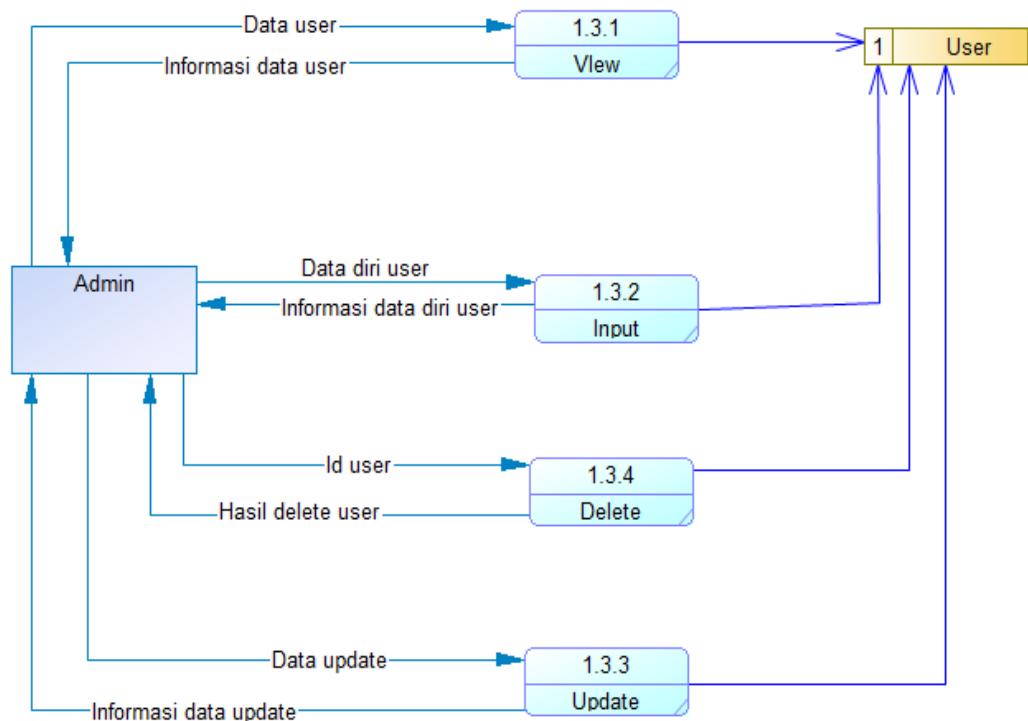


Gambar 5. 8 Data Flow Diagram Level 2 Aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital*.

Tabel 5. 9 Keterangan Data *Flow Diagram* Level 2 aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital* (Proses keseluruhan).

No Uji	Proses	Masukan	Keluaran
1.1	<i>Login</i>	<i>Input Username dan Password</i>	<i>Validasi Login</i>
1.2	<i>Home</i>	<i>Drop</i>	<i>Drag</i>
1.3	<i>Kelola Data Users</i>	<i>Insert data Users, Update data Users, Delete data Users</i>	<i>View data Users, View Update data Users, View Delete data Users</i>
1.4	<i>Ticket</i>	<i>Insert ticket, Update ticket, Delete ticket</i>	<i>View data ticket, View update data ticket, View Delete data ticket</i>
1.5	<i>Department</i>	<i>Insert Department, Update Department, Delete Department</i>	<i>View data department, View update data department, View Delete data Department</i>

5.2.2.2 Data Flow Diagram Level 2 Proses 1 Aplikasi Sistem Monitoring Pada Job Desk Human Capital



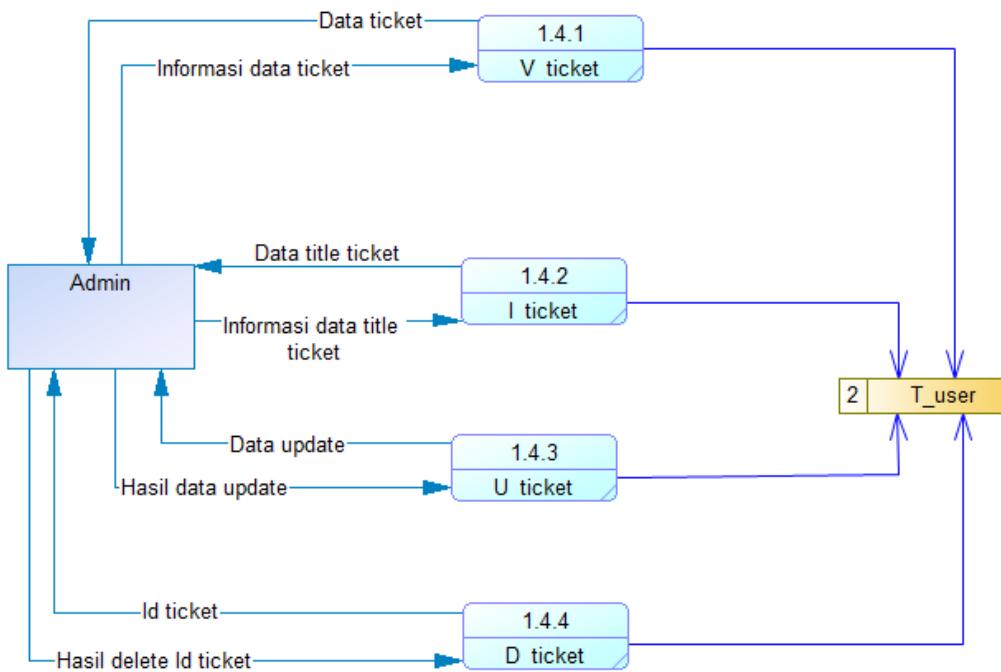
Gambar 5. 9 Data Flow Diagram Level 2 Proses 1 Kelola Data Users

Tabel 5. 10 Keterangan Data Flow Diagram Level 2 Proses 1 Kelola Data Users

No Uji	Proses	Masukan	Keluaran
1.3.1	<i>View</i>	<i>Data Users</i>	<i>Informasi data Users</i>
1.3.2	<i>Input</i>	<i>Data diri Users</i>	<i>Informasi data diri Users</i>

1.3.3	<i>Update</i>	<i>Data Update</i>	Informasi <i>data Update</i>
1.3.4	<i>Delete</i>	<i>Id Users</i>	Hasil <i>Delete Users</i>

5.2.2.3 Data Flow Diagram Level 2 Proses 2 Aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital*

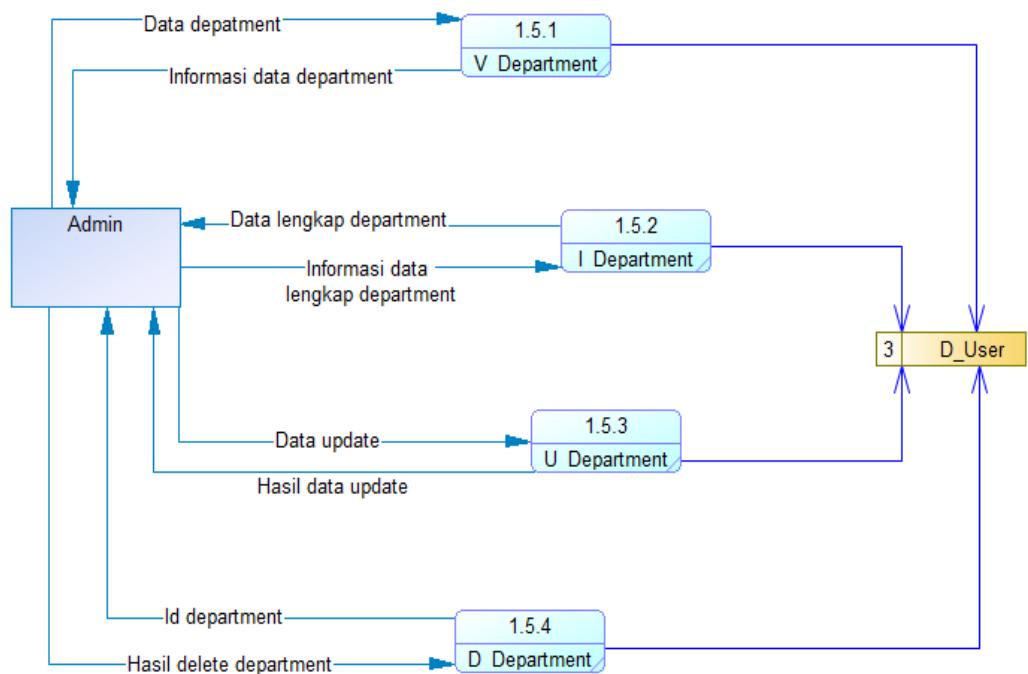


Gambar 5. 10 Data Flow Diagram Level 2 Proses 2 Kelola Data Ticket

Tabel 5. 11 Keterangan Data *Flow Diagram* Level 2 Proses 2 Kelola Data *Ticket*

No Uji	Proses	Masukan	Keluaran
1.4.1	<i>View ticket</i>	<i>Data ticket</i>	Informasi <i>data ticket</i>
1.4.2	<i>Input ticket</i>	<i>Data title ticket</i>	Informasi data title ticket
1.3.3	<i>Update ticket</i>	<i>Data Update</i>	Hasil <i>data Update</i>
1.3.4	<i>Delete ticket</i>	<i>Id ticket</i>	Hasil <i>Delete id ticket</i>

5.2.2.4 Data *Flow Diagram* Level 2 Proses 3 Aplikasi Sistem *Monitoring* Pada *Job Desk Human Capital*



Gambar 5. 11 Data *Flow Diagram* Level 2 Proses 3 *Kelola Data Department*

Tabel 5. 12 Keterangan *Data Flow Diagram* Level 2 Proses 3 Kelola Data *Department*

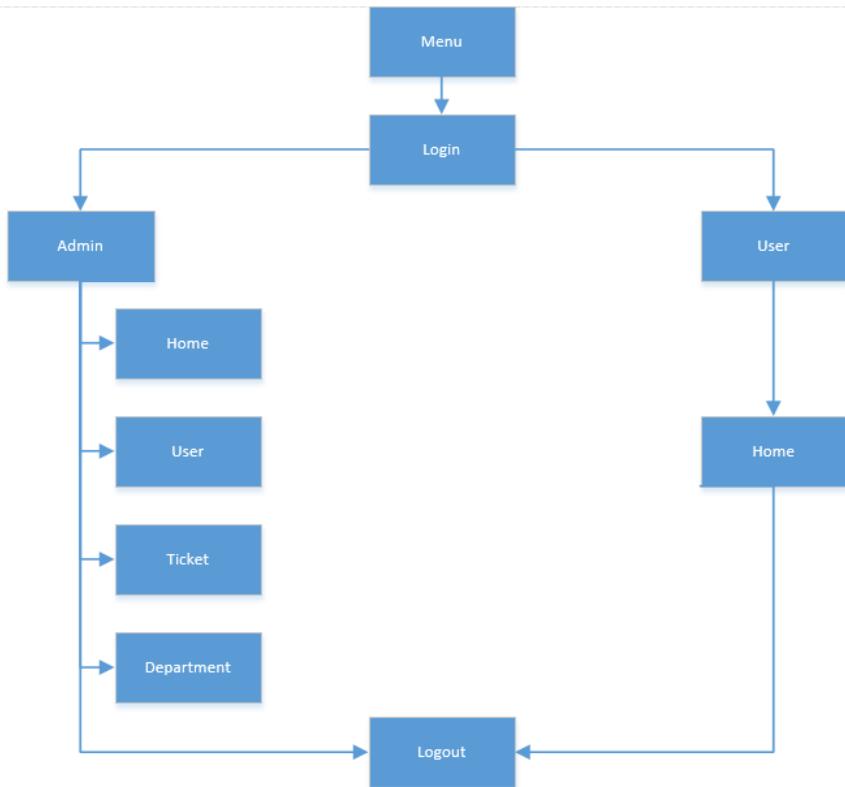
No Uji	Proses	Masukan	Keluaran
1.5.1	<i>View Department</i>	<i>Data Department</i>	Informasi <i>data Department</i>
1.5.2	<i>Input Department</i>	<i>Data lengkap Department</i>	Informasi data lengkap <i>Department</i>
1.5.3	<i>Update Department</i>	<i>Data Update</i>	Hasil <i>data Update</i>
1.5.4	<i>Delete Department</i>	<i>Id Department</i>	Hasil <i>Delete Department</i>

5.2.3 Kamus Alur Data

Tabel 5. 13 Kamus Alur Data

Nama Aliran Data	Aliran Data
Data <i>Login Admin</i>	[<i>Username Admin + Password Admin</i>]
Konfirmasi <i>Login Admin</i>	/*Informasi untuk memberitahukan bahwa <i>Login Admin</i> benar atau salah
Data <i>Users</i>	[<i>nama Users + Username + nama + email + level + aksi</i>]
Informasi <i>Data Users</i>	/*Informasi untuk menampilkan data <i>Users</i>
Data <i>ticket</i>	[<i>assign to + title + description + File choose + deadline</i>]
Informasi <i>data ticket</i>	/*Informasi untuk memberitahukan <i>data ticket</i>
<i>Department</i>	/*Informasi untk memberitahukan data <i>Department</i>
Data <i>Department</i>	[<i>no.Department + Department + aksi Department</i>]
Data <i>Users</i>	[<i>no. anggota + Username + jenis Login +status</i>]

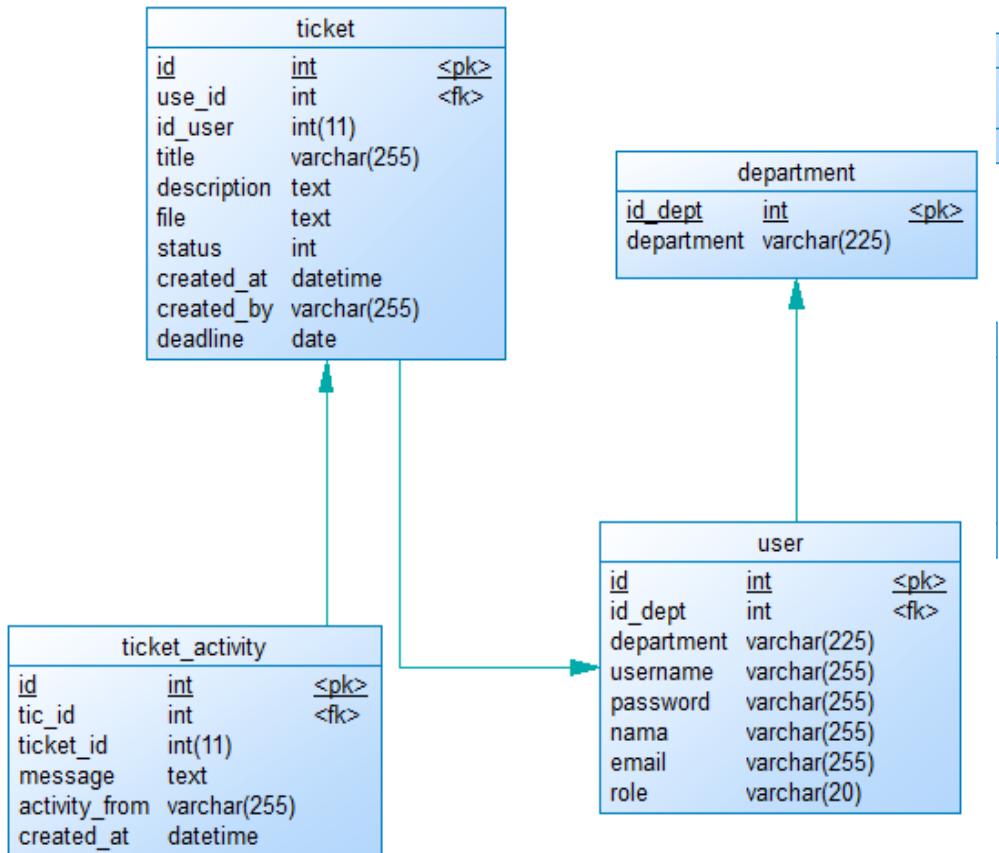
5.2.4 Struktur Menu



Gambar 5. 12 Struktur *Menu* Sistem Monitoring Terhadap *Job Desk Human Capital*

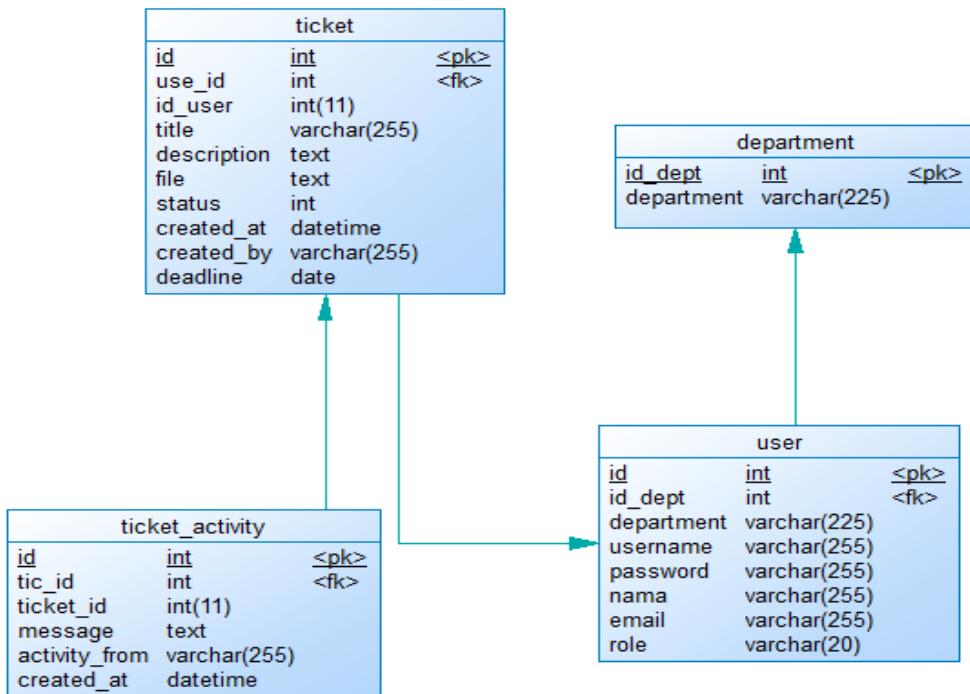
5.3 PERANCANGAN BASIS DATA

5.3.1 Conceptual Data Model (CDM)



Gambar 5. 13 *Conceptual Data Model Sistem Monitoring Terhadap Job desc Human Capital)*

5.3.2 Physical Data Model (PDM)



Gambar 5. 14 *Physical Data Model Model Sistem Monitoring Terhadap Job desc Human Capital*

5.4 MEMERSIAPKAN *TOOLS*

Untuk membuat SISTEM *MONITORING* TERHADAP *JOB DESK OPERATIONAL HUMAN*, penulis memerlukan beberapa *tools* yang dapat mendukung dalam pembuatannya dan *tools* ini sudah dibahas pada bab sebelumnya, berikut *tools* yang harus dipersiapkan dalam pembuatan sistem:

1. *Sublime Text 3*
2. Bahasa pemrograman *HyperText Presprocessor (PHP)*
3. *XAMPP*

5.4.1 Membuat *Database*

Dalam memulai pembuatan *Database*, ada beberapa step diantaranya:

1. Buka *PHPMyAdmin*, kemudian buatlah *Database* baru dengan nama “*db_ohc*”

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows recent databases: db, information_schema, mysql, performance_schema, phpmyadmin, test, and webauth.
- Top navigation bar:** Includes links for Databases, SQL, Status, Users, Export, Import, Settings, Replication, Variables,Charsets, and Engines. The "Databases" tab is selected.
- Main area - Databases:**
 - A "Create database" input field contains "db_ohc".
 - A "Collation" dropdown is set to "latin1_swedish_ci".
 - A "Create" button is visible.
 - A note at the bottom says: "Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server."
 - A list of existing databases is shown:

Database	Collation	Check Privileges
db_ohc	latin1_swedish_ci	Check Privileges
db_riki	latin1_swedish_ci	Check Privileges
information_schema	utf8_general_ci	Check Privileges
mysql	latin1_swedish_ci	Check Privileges
performance_schema	utf8_general_ci	Check Privileges
phpmyadmin	utf8_bin	Check Privileges
test	latin1_swedish_ci	Check Privileges
webauth	latin1_general_ci	Check Privileges
 - Total count: 8
 - Action buttons: Check All, With selected, Drop.
 - Enable Statistics checkbox.

Gambar 5. 15 Database db_ohc

- Setelah itu, buat tabel “departmen” dengan 2 kolom. Tabel ini nanti akan menyimpan data nama *Department* atau data *Menu* list *Department*.

Kolom yang dibutuhkan :

- Id, bertipe data integer dengan panjang 11(*Primary Key*)
- Department*, bertipe data *Varchar* dengan panjang 225

The screenshot shows the MySQL Workbench interface for the 'db_ohc' database. The 'department' table is selected. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	<code>id_dept</code>	<code>int(11)</code>			No	<code>None</code>	<code>AUTO_INCREMENT</code>	Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	<code>department</code>	<code>varchar(225)</code>	<code>latin1_swedish_ci</code>		No	<code>None</code>		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Below the table, there are buttons for 'Print view', 'Relation view', 'Propose table structure', 'Track table', and 'Move columns'. A 'Add' button with a dropdown menu for adding columns (1, column(s), At End of Table, At Beginning of Table, After id_dept) and a 'Go' button are also present.

The 'Information' tab is selected, displaying the following details:

Space usage		Row statistics	
Data	16 Kib	Format	Compact
Index	0 B	Collation	<code>latin1_swedish_ci</code>
Total	16 Kib	Next autoindex	7
		Creation	Jan 01, 2020 at 01:45 PM

Gambar 5. 16 Tabel *Department*

- Setelah itu, buat tabel “*ticket*” dengan 9 kolom. Tabel ini nanti akan menyimpan data pekerjaan yang akan dikirim

Kolom yang dibutuhkan :

- id, bertipe data *integer* dengan panjang 11 (*Primary Key*)
- id_Users*, bertipe data *Integer* dengan panjang 15
- title*, bertipe data *varchar* dengan panjang 255

- *description*, bertipe data *Text (Default Length)*
- *File*, bertipe data *Text (Default Length)*
- *status*, bertipe data *integer* dengan panjang 11
- *created_at*, bertipe data *datetime* (untuk tanggal)
- *created_by*, bertipe data *varchar* dengan panjang 255
- *deadline*, bertipe data *datetime* (untuk tanggal)

The screenshot shows the MySQL Workbench interface with the 'ticket' table selected. The table has 9 columns:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)	latin1_swedish_ci	No	None	AUTO_INCREMENT		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	id_user	int(11)	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	title	varchar(255)	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	description	text	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values
5	file	text	latin1_swedish_ci	Yes	NULL			Change Drop Primary Unique Index Spatial Fulltext Distinct values
6	status	int(11)	latin1_swedish_ci	Yes	0			Change Drop Primary Unique Index Spatial Fulltext Distinct values
7	created_at	datetime	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values
8	created_by	varchar(255)	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values
9	deadline	datetime	latin1_swedish_ci	No	None			Change Drop Primary Unique Index Spatial Fulltext Distinct values

Below the table, there are buttons for 'Check All', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', 'Fulltext', and 'Distinct values'. There are also links for 'Print view', 'Relation view', 'Propose table structure', 'Track table', and 'Move columns'. A 'Add' button with a value of 1, a 'column(s)' dropdown, and a 'At End of Table' radio button are visible. The 'Information' tab is selected, showing 'Space usage' (Data: 16 KiB, Index: 0 B, Total: 16 KiB) and 'Row statistics' (Format: Compact, Collation: latin1_swedish_ci, Next autoindex: 22, Creation: Jan 01, 2020 at 01:45 PM).

Gambar 5. 17 Tabel *ticket*

- Setelah itu, buat tabel “*ticket_activity*” dengan 5 kolom.

Kolom yang dibutuhkan:

- *id*, bertipe data *integer* dengan panjang 11
- *ticket_id*, bertipe data *integer* dengan panjang 11
- *message*, bertipe data *Text (Default Length)*
- *activity_from* bertipe data *Varchar* dengan panjang 255
- *created_at* bertipe data *datetime* (untuk tanggal)

The screenshot shows the MySQL Workbench interface for a database named 'db_niko'. The current table is 'ticket_activity'. The top navigation bar includes 'Browse', 'Structure' (selected), 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'Triggers'. Below the navigation is a table structure with columns: #, Name, Type, Collation, Attributes, Null, Default, Extra, and Action. The columns listed are:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)			No	AUTO_INCREMENT		Primary Index Fulltext
2	ticket_id	int(11)			No	None		Primary Index Fulltext
3	message	text	latin1_swedish_ci		No	None		Primary Index Fulltext
4	activity_from	varchar(255)	latin1_swedish_ci		No	None		Primary Index Fulltext
5	created_at	datetime			No	None		Primary Index Fulltext

Below the table structure, there are buttons for 'Check All', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', and 'Index'. Further down are links for 'Print view', 'Relation view', 'Propose table structure', 'Track table', and 'Move columns'. A 'Add' button with a dropdown menu (set to '1 column(s)') and a 'Go' button are also present. The 'Indexes' section is collapsed.

Information

Space usage		Row statistics	
Data	16 kB	Format	Compact
Index	0 B	Collation	latin1_swedish_ci
Total	16 kB	Next autoindex	20
		Creation	Jan 01, 2020 at 01:45 PM

Gambar 5. 18 Tabel “*ticket_activity*”

- Setelah itu, buat tabel “*Users*” dengan 7 kolom. Tabel ini nanti akan menyimpan data nama karyawan.

Kolom yang digunakan:

- id, bertipe data *integer* dengan panjang 11
- *Department*, bertipe data *varchar* dengan panjang 255
- *Usersname*, bertipe data *varchar* dengan panjang 255
- *Password*, bertipe data *varchar* dengan panjang 255
- nama, bertipe data *varchar* dengan panjang 255
- *email*, bertipe data *varchar* dengan panjang 255
- role, bertipe data *varchar* dengan panjang 20

Server: 127.0.0.1 » Database: db_niko » Table: user

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)	latin1_swedish_ci	No	None	AUTO_INCREMENT		
2	department	varchar(255)	latin1_swedish_ci	Yes	NULL			
3	username	varchar(255)	latin1_swedish_ci	No	None			
4	password	varchar(255)	latin1_swedish_ci	No	None			
5	nama	varchar(255)	latin1_swedish_ci	No	None			
6	email	varchar(255)	latin1_swedish_ci	No	None			
7	role	varchar(20)	latin1_swedish_ci	No	user			

With selected:

Add 1 column(s) At End of Table At Beginning of Table After id

+ Indexes

Information

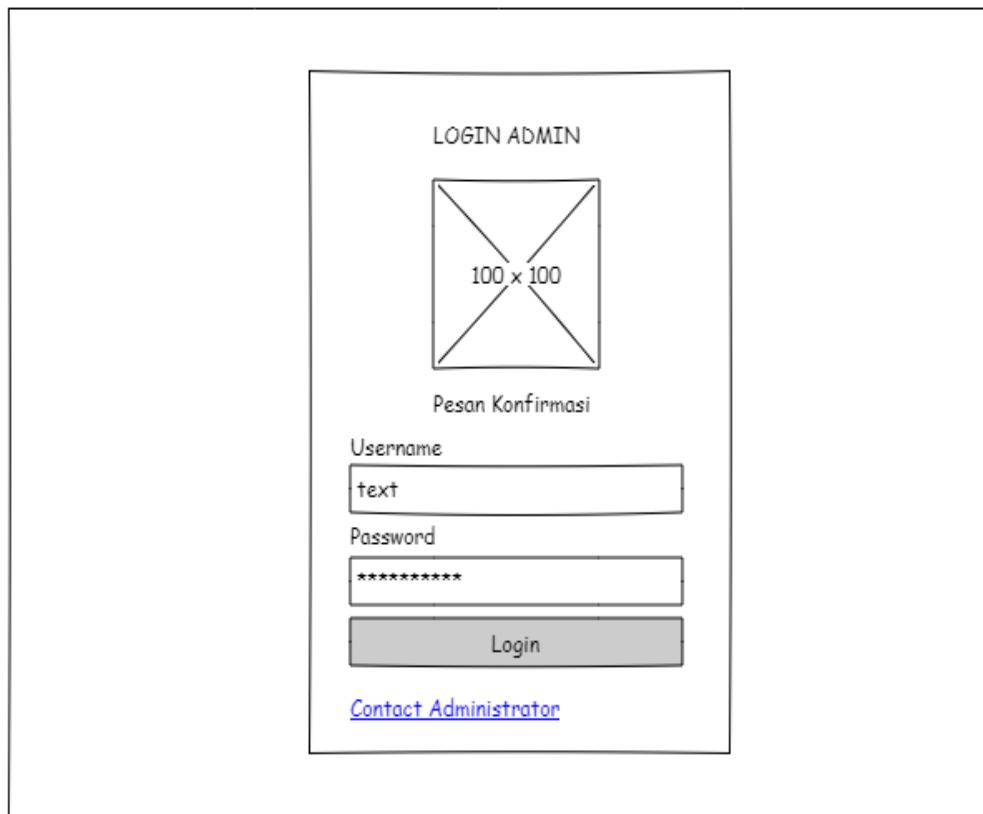
Space usage	Row statistics
Data	16 KiB
Index	0 B
Total	16 KiB
Format	
Collation	latin1_swedish_ci
Next autoindex	14
Creation	Jan 01, 2020 at 01:45 PM

Gambar 5. 19 Tabel Users

5.5 PERANCANGAN ANTARMUKA

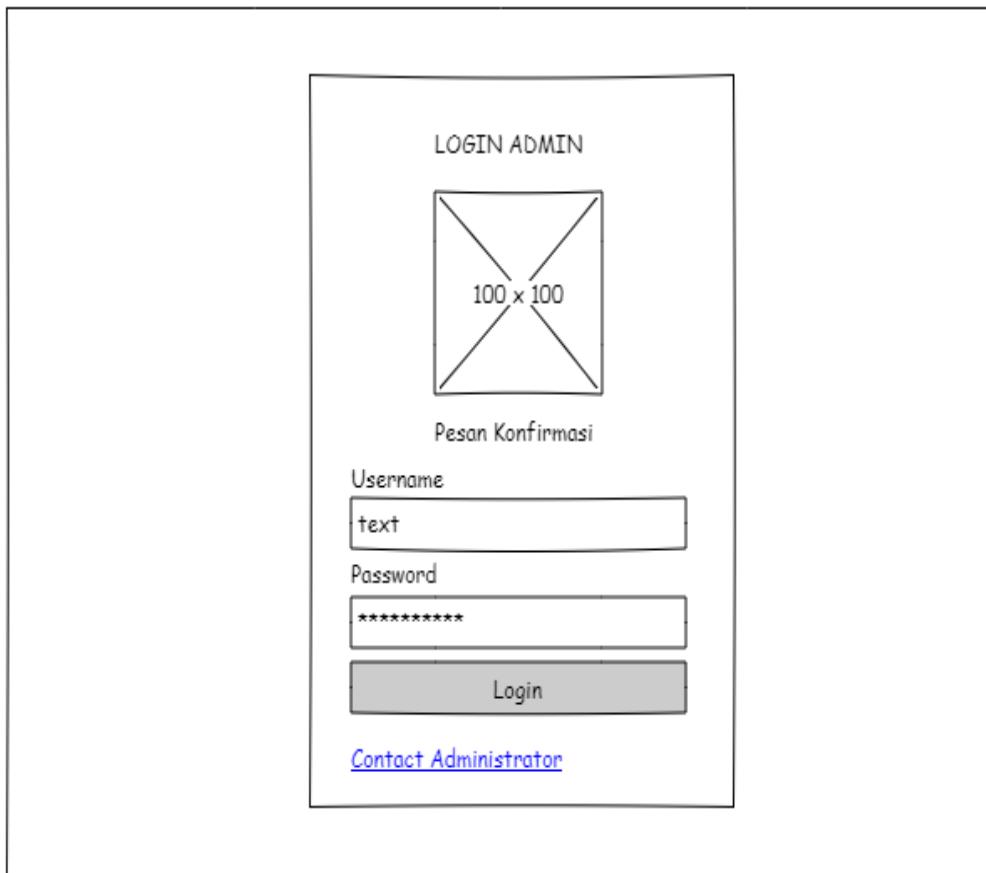
5.5.1 Perancangan Antarmuka *Login*

5.5.1.1 Perancangan Antarmuka *Login Admin*



Gambar 5. 20 Antarmuka *Login Admin*

5.5.1.2 Perancangan Antarmuka *Login Users*



Gambar 5. 21 Antarmuka *Login Users*

Berikut *Source Code* untuk halaman *Login* :

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <link rel="apple-touch-icon" sizes="76x76"
        href="assets/img/logo/pegadaian.png">
```

```

<link rel="icon" Type="image/png" href="assets/img/logo/pegadaian.png">
<meta http-equiv="X-UA-Compatible" content="IE=edge,Chrome=1" />
<title>
    Login |Support Ticket
</title>
<meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0,
Users-scalable=0, shrink-to-fit=no' name='Viewport' />
<meta property="image" content="assets/img/logo/pegadaian.png">
<meta name="twitter:image" property="og.image"
content="assets/img/logo/pegadaian.png">
<link rel="shortcut icon" Type="image/png"
href="assets/img/logo/pegadaian.png">
<!-- Fonts and icons -->
<link
href="https://fonts.googleapis.com/css?family=Montserrat:400,700,200"
rel="stylesheet" />
<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">
<!-- CSS Files -->
<link href="assets/css/bootstrap.min.css" rel="stylesheet" />
<link href="assets/css/paper-Dashboard.css?v=2.0.0" rel="stylesheet" />
<!-- CSS Just for demo purpose, don't include it in your project -->
<link href="assets/demo/demo.css" rel="stylesheet" />
</head>

<body style="background-image: url('assets/img/logo/background-
pegadaian.png'); background-repeat: no-repeat; background-position: center
center;background-size: cover;padding: 0; margin-top: 5px">
```

```
<div class="wrapper">
    <section class="Login_content">
        <form action="logedin.PHP" method="POST">
            <div class="Text-center">
                
            </div>
            <div class="form-group">
                <input Type="Text" name="Usersname" class="form-control" placeholder="Usersname" required="" />
            </div>
            <div class="form-group">
                <input Type="Password" name="Password" class="form-control" placeholder="Password" required="" />
            </div>
            <div class="form-group">
                <center>
                    <input Type="submit" class="btn btn-success btn-block submit" value="Log in">
                </center>
            </div>
        <div class="clearfix"></div>

        <div class="separator">
            <div class="clearfix"></div>

```

```
<div class="Text-center">  
    <p>©<?= date('Y') ?> All Rights Reserved. Privacy and Terms</p>  
</div>  
</div>  
</form>  
</section>  
</div>  
<style>  
body {  
    background-color: #f0f0f0;  
}  
.Login_content {  
    background-color: #fff;  
    border-radius: 10px;  
    position: absolute;  
    top: 100px;  
    left: 0;  
    right: 0;  
    margin: 0 auto;  
    width: 400px;  
    padding: 10px 30px;  
    -webkit-box-shadow: 0px 6px 13px -1px rgba(221,221,221,1);  
    -moz-box-shadow: 0px 6px 13px -1px rgba(221,221,221,1);  
    box-shadow: 0px 6px 13px -1px rgba(221,221,221,1);  
}
```

```
</style>

<!-- Core JS Files -->

<script src="assets/js/core/jquery.min.js"></script>
<script src="assets/js/core/popper.min.js"></script>
<script src="assets/js/core/bootstrap.min.js"></script>
<script src="assets/js/plugins/perfect-scrollbar.jquery.min.js"></script>

<!-- Chart JS -->

<script src="assets/js/plugins/chartjs.min.js"></script>

<!-- Notifications Plugin -->

<script src="assets/js/plugins/bootstrap-notify.js"></script>

<!-- Control Center for Now Ui Dashboard: parallax effects, scripts for the
example pages etc -->

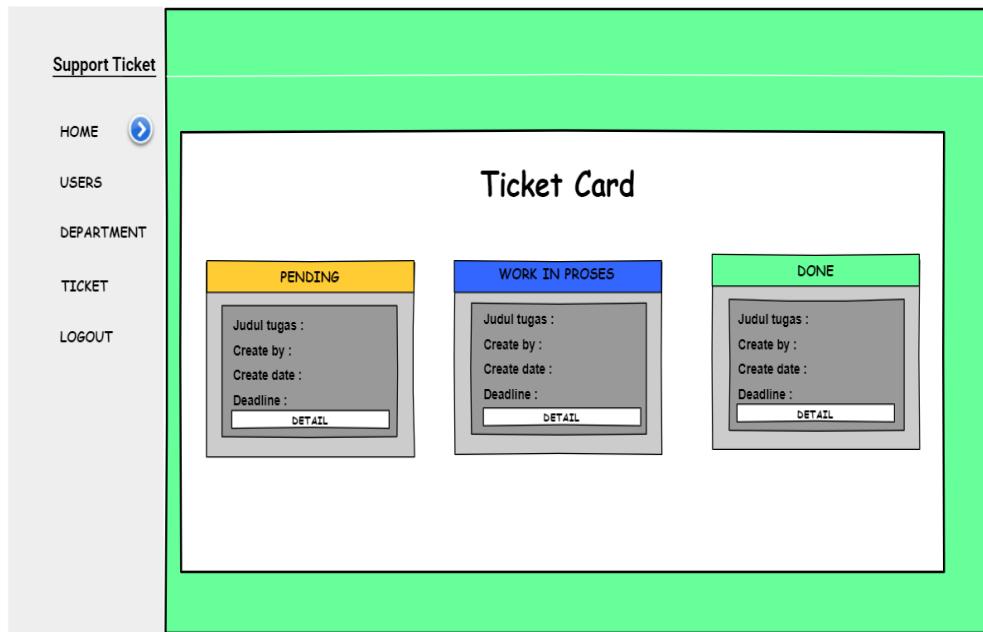
<script src="assets/js/paper-Dashboard.min.js?v=2.0.0"
Type="Text/javascript"></script>

</body>

</html>
```

5.5.2 Perancangan Antarmuka *Dashboard*

5.5.2.1 Perancangan Antarmuka *Dashboard Admin*



Gambar 5. 22 *Antarmuka Dashboard Admin*

Berikut *Source Code* untuk halaman *Dashboard Admin* :

```
<link rel="stylesheet" Type="Text/css" href="assets/css/card.css">
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card Text-center">
<div class="card-header ">
<h5 class="card-title">Ticket Card</h5>
</div>
```

```

<div class="card-body ">
  <div class="drag-container">
    <ul class="drag-list">
      <li class="drag-column drag-column-on-hold">
        <span class="drag-column-header">
          <h2 align="center">Pending</h2>
        </span>

      <ul class="drag-inner-list" id="1" value="1">
        <?PHP
          if ($_SESSION['role'] == 'Admin') {
            $pending = $db->query("SELECT * FROM ticket WHERE
status=1 ORDER BY created_at DESC", 0);
          } else{
            $pending = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=1 ORDER BY created_at
DESC", 0);
          }
        while($row_pending = $pending->fetch_assoc()) {
          ?>
          <li class="drag-item" value="<?= $row_pending['id'] ?>">
            <div class="title"><?= $row_pending['title'] ?></div>
            <div class="created_by">Created By : <span><?=
$row_pending['created_by'] ?></span></div>
            <div class="date">Created Date : <span><?= $helpers-
>dateTimeIndonesia($row_pending['created_at']) ?></span></div>

```

```

<div class="date">Deadline : <span><?= $helpers->dateTimeIndonesia($row_pending['created_at']) ?></span></div>

    <button Type="button" class="button" data-toggle="modal" data-target="#detail" onclick="getDetail(<?= $row_pending['id'] ?>)">

        DETAIL

    </button>

</li>

<?PHP } ?>

</ul>

</li>

<li class="drag-column drag-column-in-progress">

    <span class="drag-column-header">

        <h2 align="center">Work in Progress</h2>

    </span>

    <ul class="drag-inner-list" id="2" value="0">

        <?PHP

            if ($_SESSION['role'] == 'Admin') {

                $progress = $db->query("SELECT * FROM ticket WHERE
status=0 ORDER BY created_at DESC", 0);

            }else{

                $progress = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=0 ORDER BY created_at
DESC", 0);

            }

            while($row_progess = $progress->fetch_assoc()) {

                ?>

```

```

<li class="drag-item" value=<?= $row_progess['id'] ?>>
    <div class="title"><?= $row_progess['title'] ?></div>
    <div class="created_by">Created By : <span><?= $row_progess['created_by'] ?></span></div>
    <div class="date">Created Date : <span><?= $helpers->dateTimeIndonesia($row_progess['created_at']) ?></span></div>
    <div class="date">Deadline : <span><?= $helpers->dateTimeIndonesia($row_progess['deadline']) ?></span></div>
    <button Type="button" class="button" data-toggle="modal" data-target="#detail" onclick="getDetail(<?= $row_progess['id'] ?>)">
        DETAIL
    </button>
</li>
<?PHP } ?>
</ul>
</li>
<li class="drag-column drag-column-approved">
    <span class="drag-column-header">
        <h2 align="center">Done</h2>
    </span>
    <ul class="drag-inner-list" id="3" value="2">
        <?PHP
            if ($_SESSION['role'] == 'Admin') {
                $done = $db->query("SELECT * FROM ticket WHERE status=2 ORDER BY created_at DESC", 0);
            } else{

```

```

$done = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=2 ORDER BY created_at
DESC", 0);

}

while($row_done = $done->fetch_assoc()) {

?>

<li class="drag-item" value="<?= $row_done['id'] ?>">
<div class="title"><?= $row_done['title'] ?></div>
<div class="created_by">Created By : <span><?= $row_done['created_by'] ?></span></div>
<div class="date">Created Date : <span><?= $helpers->dateTimeIndonesia($row_done['created_at']) ?></span></div>
<div class="date">Deadline : <span><?= $helpers->dateTimeIndonesia($row_done['deadline']) ?></span></div>
<button Type="button" class="button" data-toggle="modal"
data-target="#detail" onclick="getDetail(<?= $row_done['id'] ?>)">
    DETAIL
</button>
</li>
<?PHP } ?>
</ul>
</li>
<!-- <?PHP if ($_SESSION['role'] != 'Admin') { ?>
<li class="drag-column drag-column-approved">
<span class="drag-column-header">
<h2 align="center">Done</h2>
</span>

```

```

<ul class="drag-inner-list" id="3" value="2">
<?PHP
if ($_SESSION['role'] == 'Admin') {
    $done = $db->query("SELECT * FROM ticket WHERE
status=2 ORDER BY created_at DESC", 0);
} else{
    $done = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=2 ORDER BY created_at
DESC", 0);
}

while($row_done = $done->fetch_assoc()) {
?>
<li class="drag-item" value=<?= $row_done['id'] ?>>
    <div class="title"><?= $row_done['title'] ?></div>
    <div class="created_by">Created By : <span><?=
$row_done['created_by'] ?></span></div>
    <div class="date">Created Date : <span><?= $helpers-
>dateTimeIndonesia($row_done['created_at']) ?></span></div>
    <button Type="button" class="button" data-toggle="modal"
data-target="#detail" onclick="getDetail(<?= $row_done['id'] ?>)">
        DETAIL
    </button>
</li>
<?PHP } ?>
</ul>
</li>
<?PHP } ?> -->

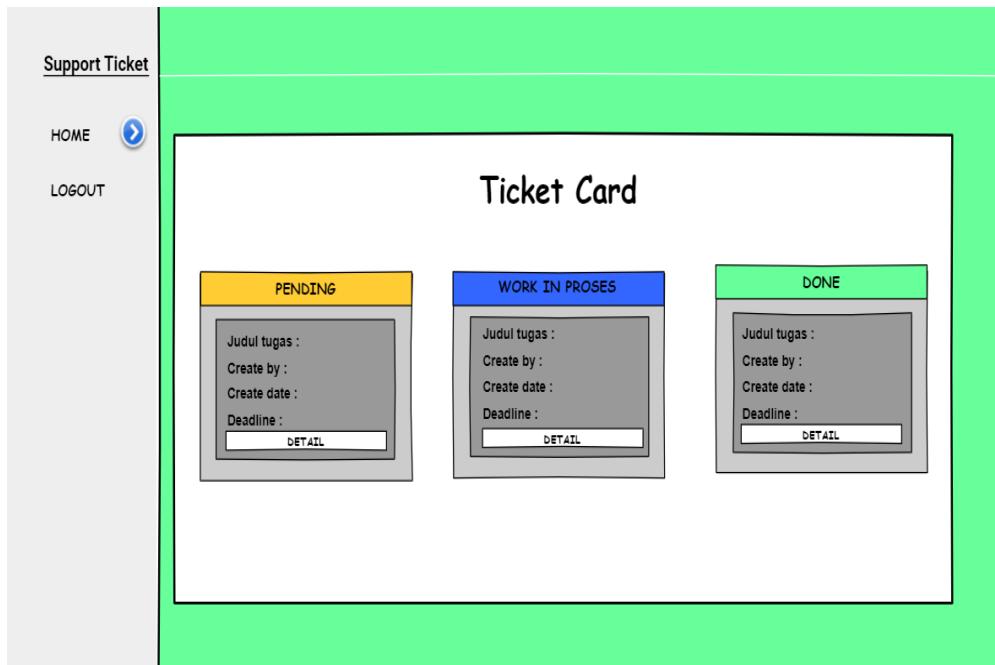
```

```
</ul>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="modal fade" id="detail" tabindex="-1" role="dialog" aria-labelledby="label" aria-hidden="true">
<div class="modal-dialog modal-dialog-centered" role="document">
<div id="modal-content" class="modal-content"></div>
</div>
</div>

<script src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/45226/dragula.min.js"></script>
<script src="assets/js/card.js"></script>
```

5.5.2.2 Perancangan Antarmuka *Dashboard Users*



Gambar 5. 23 Antarmuka *Dashboard Users*

Berikut *Source Code* untuk halaman *Dashboard Users* :

```
<link rel="stylesheet" Type="Text/css" href="assets/css/card.css">

<div class="content">
    <div class="row">
        <div class="col-md-12">
            <div class="card Text-center">
                <div class="card-header ">
                    <h5 class="card-title">Ticket Card</h5>
                </div>
                <div class="card-body ">
                    <div class="drag-container">
                        <ul class="drag-list">
```

```

<li class="drag-column drag-column-on-hold">
    <span class="drag-column-header">
        <h2 align="center">Pending</h2>
    </span>

    <ul class="drag-inner-list" id="1" value="1">
        <?PHP
            if ($_SESSION['role'] == 'Admin') {
                $pending = $db->query("SELECT * FROM ticket WHERE
status=1 ORDER BY created_at DESC", 0);
            } else{
                $pending = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=1 ORDER BY created_at
DESC", 0);
            }
        while($row_pending = $pending->fetch_assoc()) {
            ?>
            <li class="drag-item" value="<?= $row_pending['id'] ?>">
                <div class="title"><?= $row_pending['title'] ?></div>
                <div class="created_by">Created By : <span><?=
$row_pending['created_by'] ?></span></div>
                <div class="date">Created Date : <span><?= $helpers-
>dateTimeIndonesia($row_pending['created_at']) ?></span></div>
                <div class="date">Deadline : <span><?= $helpers-
>dateTimeIndonesia($row_pending['created_at']) ?></span></div>
                <button Type="button" class="button" data-toggle="modal"
data-target="#detail" onclick="getDetail(<?= $row_pending['id'] ?>)">

```

DETAIL

```
</button>

</li>

<?PHP } ?>

</ul>

</li>

<li class="drag-column drag-column-in-progress">

<span class="drag-column-header">

<h2 align="center">Work in Progress</h2>

</span>

<ul class="drag-inner-list" id="2" value="0">

<?PHP

if ($_SESSION['role'] == 'Admin') {

    $progress = $db->query("SELECT * FROM ticket WHERE
status=0 ORDER BY created_at DESC", 0);

} else{

    $progress = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=0 ORDER BY created_at
DESC", 0);

}

while($row_progess = $progress->fetch_assoc()) {

?>

<li class="drag-item" value="<?= $row_progess['id'] ?>">

<div class="title"><?= $row_progess['title'] ?></div>

<div class="created_by">Created By : <span><?=
$row_progess['created_by'] ?></span></div>
```

```

<div class="date">Created Date : <span><?= $helpers->dateTimeIndonesia($row_progess['created_at']) ?></span></div>

<div class="date">Deadline : <span><?= $helpers->dateTimeIndonesia($row_progess['deadline']) ?></span></div>

<button Type="button" class="button" data-toggle="modal" data-target="#detail" onclick="getDetail(<?= $row_progess['id'] ?>)">
    DETAIL
</button>
</li>
<?PHP } ?>
</ul>
</li>
<li class="drag-column drag-column-approved">
    <span class="drag-column-header">
        <h2 align="center">Done</h2>
    </span>
    <ul class="drag-inner-list" id="3" value="2">
        <?PHP
            if ($_SESSION['role'] == 'Admin') {
                $done = $db->query("SELECT * FROM ticket WHERE status=2 ORDER BY created_at DESC", 0);
            }else{
                $done = $db->query("SELECT * FROM ticket WHERE id_Users=".$_SESSION['id']."' AND status=2 ORDER BY created_at DESC", 0);
            }
        while($row_done = $done->fetch_assoc()) {

```

```

?>

<li class="drag-item" value="<?= $row_done['id'] ?>">
    <div class="title"><?= $row_done['title'] ?></div>
    <div class="created_by">Created By : <span><?= $row_done['created_by'] ?></span></div>
    <div class="date">Created Date : <span><?= $helpers->dateTimeIndonesia($row_done['created_at']) ?></span></div>
    <div class="date">Deadline : <span><?= $helpers->dateTimeIndonesia($row_done['deadline']) ?></span></div>
    <button Type="button" class="button" data-toggle="modal" data-target="#detail" onclick="getDetail(<?= $row_done['id'] ?>)">
        DETAIL
    </button>
</li>
<?PHP } ?>
</ul>
</li>
<!-- <?PHP if ($_SESSION['role'] != 'Admin') { ?>
<li class="drag-column drag-column-approved">
    <span class="drag-column-header">
        <h2 align="center">Done</h2>
    </span>
    <ul class="drag-inner-list" id="3" value="2">
        <?PHP
            if ($_SESSION['role'] == 'Admin') {
                $done = $db->query("SELECT * FROM ticket WHERE status=2 ORDER BY created_at DESC", 0);

```

```

}else{
    $done = $db->query("SELECT * FROM ticket WHERE
id_Users=".$_SESSION['id']."' AND status=2 ORDER BY created_at
DESC", 0);
}

while($row_done = $done->fetch_assoc()) {
?>
<li class="drag-item" value=<?= $row_done['id'] ?>">
    <div class="title"><?= $row_done['title'] ?></div>
    <div class="created_by">Created By : <span><?=
$row_done['created_by'] ?></span></div>
    <div class="date">Created Date : <span><?= $helpers-
>dateTimeIndonesia($row_done['created_at']) ?></span></div>
    <button Type="button" class="button" data-toggle="modal"
data-target="#detail" onclick="getDetail(<?= $row_done['id'] ?>)">
        DETAIL
    </button>
</li>
<?PHP } ?>
</ul>
</li>
<?PHP } ?> -->
</ul>
</div>
</div>
</div>

```

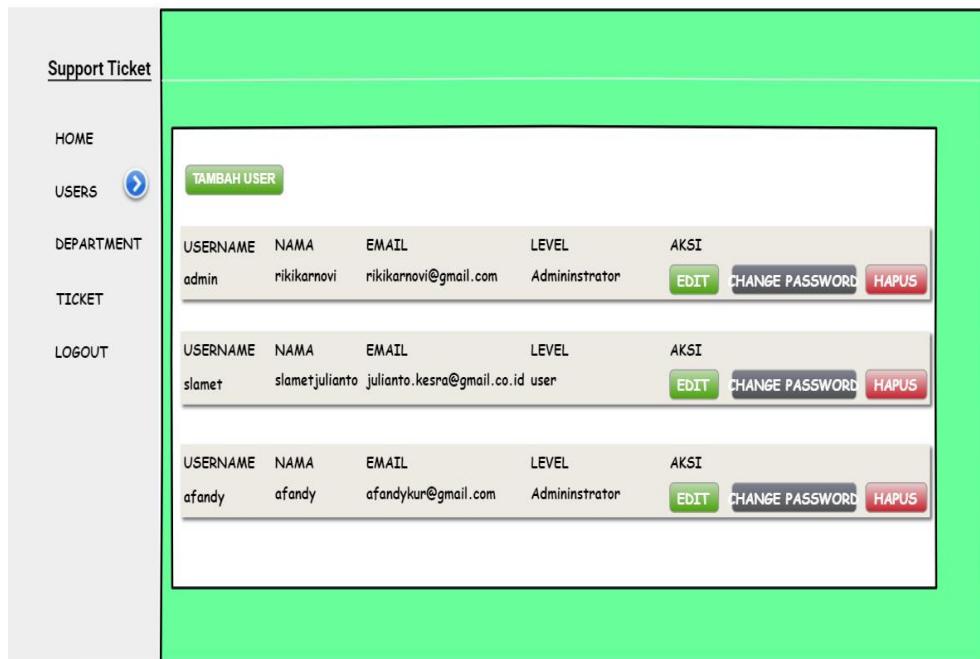
```
</div>
</div>
</div>

<div class="modal fade" id="detail" tabindex="-1" role="dialog" aria-
labelledby="label" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div id="modal-content" class="modal-content"></div>
  </div>
</div>

<script src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/45226/dragula.min.js"></script>
<script src="assets/js/card.js"></script>
```

5.5.3 Perancangan Antarmuka *Users*

5.5.3.1 Perancangan Antarmuka *View Users*



Gambar 5. 24 Antarmuka Data *Users*

Berikut *Source Code* untuk halaman Data *Users* :

```
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card ">
<div class="card-header ">
<h5 class="card-title">Users</h5>
</div>
```

```

<div class="card-body ">

    <a href="index.PHP?m=Users&s=Users_add" class="btn btn-primary">Tambah Users</a>

    <div class="table-responsive">

        <table class="table table-hover">

            <thead>

                <tr>

                    <th>Usersname</th>

                    <th>Nama</th>

                    <th>Email</th>

                    <th>Department</th>

                    <th>Level</th>

                    <th>Aksi</th>

                </tr>

            </thead>

            <tbody>

                <?PHP

                    $data = $db->query("SELECT a.id,
a.Usersname,a.nama,a.email, a.role,
b.id_dept, b.Department
FROM Users a JOIN Department b

```

```
ON a.Department=b.Department ORDER BY
role ASC, nama ASC", 0);
```

```
while($row = $data->fetch_assoc()) {  
?>  
<tr>  
<td><?= $row['Usersname'] ?></td>  
<td><?= $row['nama'] ?></td>  
<td><?= $row['email'] ?></td>  
<td><?= $row['Department'] ?></td>  
<td><?= $row['role'] == 'Admin' ? 'Administrator' : 'Users'  
?></td>  
<td>  
<a href="index.PHP?m=Users&s=Users_edit&id=<?=$row['id'] ?>" class="btn btn-xs btn-primary">Edit</a>  
<a href="index.PHP?m=Users&s=Users_Password&id=<?=$row['id'] ?>" class="btn btn-xs btn-secondary">Change Password</a>  
<a onclick="DeleteData(<?= $row['id'] ?>)" class="btn btn-xs btn-danger" style="color:#fff;cursor:pointer">Hapus</a>  
</td>  
</tr>
```

```
<?PHP } ?>

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

</script>

function DeleteData(id) {

    var r = confirm("Anda yakin ingin menghapus ini");

    if (r == true) {

        location.href = "pages/Users/Users_proses.PHP?aksi=hapus&id=" + id;

    }

}

</script>
```

5.5.3.2 Perancangan Antarmuka Add Data Users

The screenshot shows a user interface for adding a new user. On the left, there is a vertical navigation menu with options: HOME, USERS (which is selected, indicated by a blue circular icon with a white arrow), DEPARTMENT, TICKET, and LOGOUT. The main content area has a green header bar with the text 'Tambah User'. Below the header, there are five input fields labeled 'Nama', 'Username', 'Password', 'Email', and 'Department'. Each input field has a corresponding placeholder text ('Nama', 'Username', 'Password', 'Email', and 'Pilih Department'). There are also two dropdown menus labeled 'Role' and 'Department' with placeholder text 'Pilih Role' and 'Pilih Department' respectively. At the bottom of the form is a green 'SUBMIT' button.

Gambar 5. 25 Antarmuka Add Data Users

Berikut *Source Code* untuk halaman Add Data Users :

```
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card ">
<div class="card-header ">
<h5 class="card-title">Tambah Users</h5>
</div>
<div class="card-body ">
```

```
<form action="pages/Users/Users_proses.PHP?aksi=insert"
method="POST">

    <div class="form-group">

        <label for="inputText3" class="col-form-label">Nama</label>

        <input id="inputText3" name="nama" Type="Text" class="form-
control">

    </div>

    <div class="form-group">

        <label for="inputText3" class="col-form-
label">Usersname</label>

        <input id="inputText3" name="Usersname" Type="Text"
class="form-control">

    </div>

    <div class="form-group">

        <label for="inputText3" class="col-form-
label">Password</label>

        <input id="inputText3" name="Password" Type="Password"
class="form-control">

    </div>

    <div class="form-group">

        <label for="inputText3" class="col-form-label">Email</label>
```

```

<input      id="inputText3"      name="email"      Type="email"
class="form-control">

</div>

<div class="form-group">

<label        for="Department"        class="col-form-
label">Department</label>

<select name="Department" class="form-control">

<option value="">-- Pilih Department --</option>

<?PHP

//Membuat koneksi ke Database

$kon = mysqli_connect("localhost",'root','','db_riki');

if (!$kon){

die("Koneksi Database gagal:".mysqli_connect_error());

}

//Perintah sql untuk menampilkan semua data pada tabel
Department

$sql="select * from Department";

$hasil=mysqli_query($kon,$sql);

$no=0;

```

```
while ($data = mysqli_fetch_array($hasil)) {  
  
    $no++;  
  
    ?>  
  
    <option value="<?PHP echo $data['Department'];?>"><?PHP  
echo $data['Department'];?></option>  
  
    <?PHP  
  
    }  
  
    ?>  
  
</select>  
  
<div class="form-group">  
  
    <label for="role" class="col-form-label">Role</label>  
  
    <select class="form-control" name="role">  
  
        <option value="">-- Pilih Role --</option>  
  
        <option value="Admin">Administrator</option>  
  
        <option value="Users">Users</option>  
  
    </select>  
  
</div>  
  
<div class="form-group">  
  
    <input Type="submit" class="btn btn-primary" value="Submit">  
  
</div>  
  
</form>
```

```

</div>

</div>

</div>

</div>

</div>

```

5.5.3.3 Perancangan Antarmuka *Update Data Users*

The screenshot displays a user interface for updating user information. On the left, a sidebar titled 'Support Ticket' contains links for HOME, USERS (which is currently selected and highlighted with a blue arrow icon), DEPARTMENT, TICKET, and LOGOUT. The main content area is titled 'Update User' and includes fields for Username (admin), Nama (empty), Email (empty), Department (a dropdown menu showing 'Pilih Department'), and Role (a dropdown menu showing 'Pilih Role'). A green 'SUBMIT' button is located at the bottom of the form.

Gambar 5. 26 Antarmuka *Update Data Users*

Berikut *Source Code* untuk halaman *Update Data Users* :

```
<?PHP
```

```

$data    =    $db->query('SELECT    *    FROM    Users    WHERE
id="'.$_GET['id'].'");

```

```

$row = $data->fetch_assoc()

?>

<div class="content">

<div class="row">

<div class="col-md-12">

<div class="card ">

<div class="card-header ">

<h5 class="card-title">Update Users</h5>

</div>

<div class="card-body ">

<form

action="pages/Users/Users_proses.PHP?aksi=Update&id=<?= $_GET['id']"
?>" method="POST">

<div class="form-group">

<label for="inputText3" class="col-form-label">Usersname</label>

<input id="inputText3" name="Usersname" Type="Text"
value="<?= $row['Usersname'] ?>" class="form-control" readonly disabled>

</div>

<div class="form-group">

<label for="inputText3" class="col-form-label">>Nama</label>

```

```

<input id="inputText3" name="nama" Type="Text" value="<?=
$row['nama'] ?>" class="form-control">

</div>

<div class="form-group">

    <label for="inputText3" class="col-form-label">Email</label>

    <input id="inputText3" name="email" Type="email" value="<?=
$row['email'] ?>" class="form-control">

</div>

<div class="form-group">

    <label for="iddept" class="col-form-label">Department</label>

    <select name="Department" class="form-control">

        <option      value="<?=$row['Department'] ?>"><?=
$row['Department'] ?></option>

        <option value="">-- Pilih Department --</option>

    <?PHP

//Membuat koneksi ke Database

$kon = mysqli_connect("localhost",'root','','db_riki');

if (!$kon){

    die("Koneksi Database gagal:".mysqli_connect_error());

}

```

```

//Perintah sql untuk menampilkan semua data pada tabel
Department

$sql="select * from Department";

$hasil=mysqli_query($kon,$sql);

$no=0;

while ($data = mysqli_fetch_array($hasil)) {

$no++;

?>

<option value=<?PHP echo $data['Department'];?>><?PHP
echo $data['Department'];?></option>

<?PHP

}

?>

</select>

</div>

<div class="form-group">

<label for="role" class="col-form-label">Role</label>

<select class="form-control" name="role">

<option value=<?= $row['role'] == 'Admin' ? 'Administrator' :
'Users' ?><?= $row['role'] == 'Admin' ? 'Administrator' : 'Users'
?></option>

```

```
<option>-- Pilih Role --</option>

<option value="Admin" <?= $row['role'] == 'Admin' ? 'selected' : " ?>>Administrator</option>

<option value="Users" <?= $row['role'] == 'Users' ? 'selected' : " ?>>Users</option>

</select>

</div>

<div class="form-group">

    <input Type="submit" class="btn btn-primary" value="Submit">

</div>

</form>

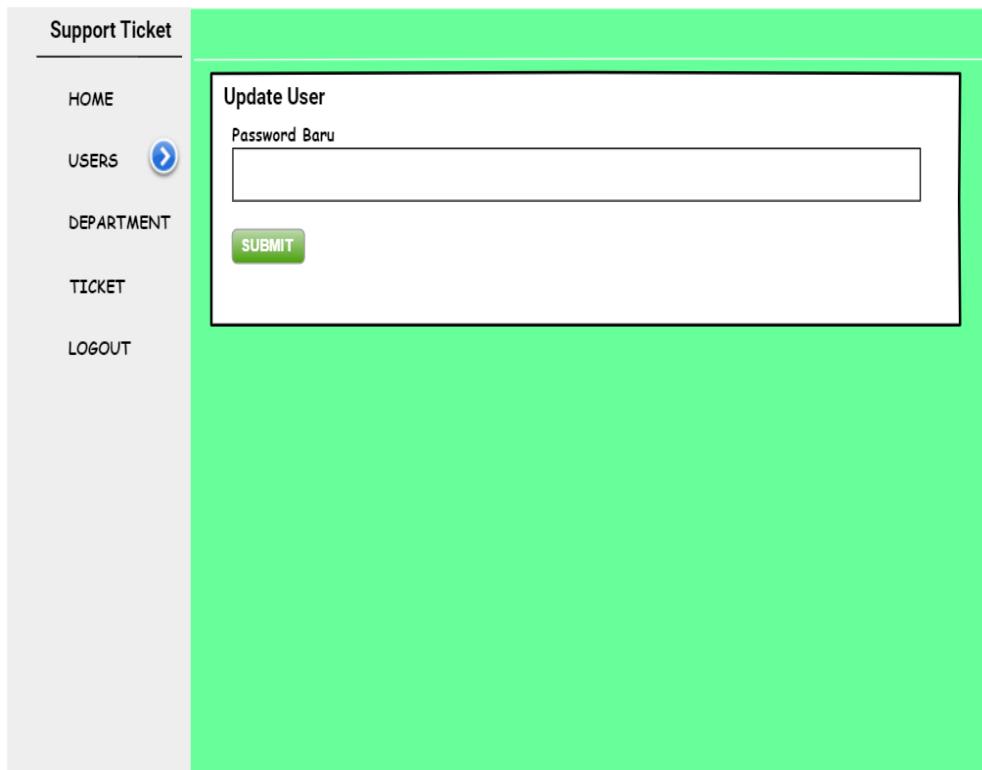
</div>

</div>

</div>

</div>
```

5.5.3.4 Perancangan Antarmuka *Change Password Users*



Gambar 5. 27 Antarmuka *Change Password Users*

Berikut *Source Code* untuk halaman *Change Password Users* :

```
<?PHP  
  
$data = $db->query('SELECT * FROM Users WHERE  
id="'. $_GET['id'].'");  
  
$row = $data->fetch_assoc()  
  
?>  
  
<div class="content">  
  
<div class="row">  
  
<div class="col-md-12">
```

```
<div class="card ">

<div class="card-header ">

<h5 class="card-title">Update Password Users</h5>

</div>

<div class="card-body ">

<form

action="pages/Users/Users_proses.PHP?aksi=Update_Password&id=<?=

$_GET['id'] ?>" method="POST">

    <div class="form-group">

        <label for="inputText3" class="col-form-label">Password

Baru</label>

        <input id="inputText3" name="Password" Type="Password"

class="form-control">

    </div>

    <div class="form-group">

        <input Type="submit" class="btn btn-primary" value="Submit">

    </div>

</form>

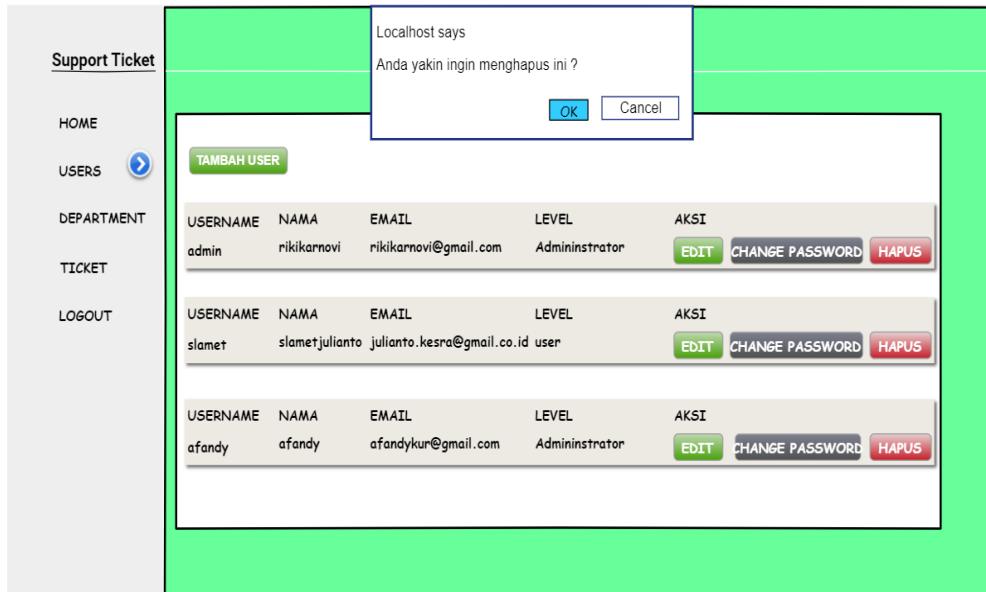
</div>

</div>

</div>
```

</div>

5.5.3.5 Perancangan Antarmuka Delete Data Users



Gambar 5. 28 Perancangan Antarmuka Delete Password Users

Berikut *Source Code* untuk halaman *Delete Password Users* :

?PHP

```
include '../../../../../db/db.PHP';
```

```
$aksi = $_GET['aksi'];
```

```
// PROSES INPUT DATA USERS
```

```
if ($aksi == 'insert') {
```

```
    $Department = $_POST['Department'];
```

```

$Usersname = $_POST['Usersname'];

$Password = md5($_POST['Password']);

$nama = $_POST['nama'];

$email = $_POST['email'];

$role = $_POST['role'];



$insert = $db->query('INSERT INTO Users (Department, Usersname,
Password, nama, email, role)

VALUES      ("'.$Department.'","'.$Usersname.'",
"'.$Password.'",
"'.$nama.'", "'.$email.'",
"'.$role.'")');




if ($insert) {

echo '<script>alert("Data berhasil ditambahkan");location.href =
"../../index.PHP?m=Users&s=Users"</script>';

} else {

echo '<script>alert("Data gagal ditambahkan");history.go(-1)</script>';

}

// PROSES DATA UPDATE USERS

} else if ($aksi == 'Update') {

```

```

$id = $_GET['id'];

$Department = $_POST['Department'];

$nama = $_POST['nama'];

$email = $_POST['email'];

$role = $_POST['role'];

// $expired = $_POST['expired_date'];




$Update = $db->query('UPDATE Users SET nama="'. $nama .'", Department="'. $Department .'", email="'. $email .'", role="'. $role .'" WHERE id="'. $id .'")';

if ($Update) {

    echo '<script>alert("Data berhasil diedit");location.href = "'.$/../index.PHP?m=Users&s=Users"</script>';

} else {

    echo '<script>alert("Data gagal diedit");history.go(-1)</script>';

}

// HAPUS DATA USERS BERDASARKAN ID

} else if ($aksi == 'hapus') {

    $id = $_GET['id'];

    $hapus = $db->query('DELETE FROM Users WHERE id="'. $id .'")';

```

```

if ($hapus) {

    echo '<script>alert("Data berhasil dihapus");location.href =
"../../index.PHP?m=Users&s=Users"</script>';

} else {

    echo '<script>alert("Data gagal dihapus");history.go(-1)</script>';

}

// PROSES CHANGE PASSWORD USERS BERDASARKAN ID

} else if ($aksi == 'Update_Password') {

$id = $_GET['id'];

$Password = md5($_POST['Password']);

$Change = $db->query('UPDATE Users SET Password="'. $Password .'"'
WHERE id="'.$id.'');

if ($Change) {

    echo '<script>alert("Password berhasil diganti");location.href =
"../../index.PHP?m=Users&s=Users"</script>';

} else {

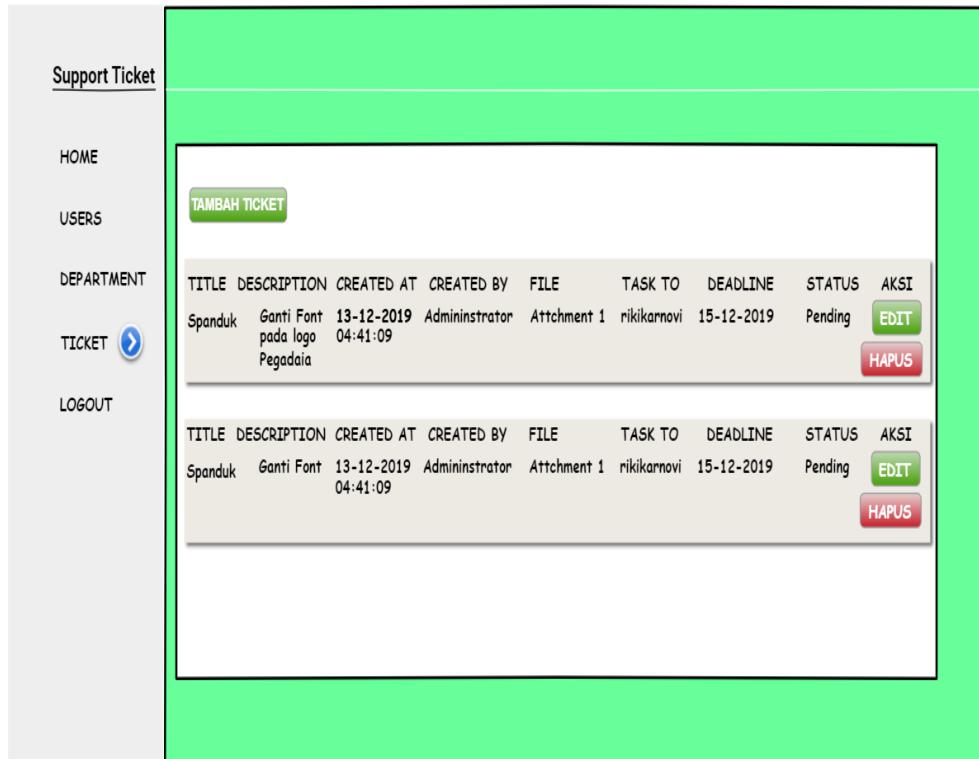
    echo '<script>alert("Password gagal diganti");history.go(-1)</script>';

}
}

```

5.5.4 Perancangan Antarmuka *Ticket*

5.5.4.1 Perancangan Antarmuka *View Data Ticket*



Gambar 5. 29 Antarmuka *View Data Ticket*

Berikut *Source Code* untuk halaman *View Data Ticket* :

```
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card ">
<div class="card-header ">
<h5 class="card-title">Ticket</h5>
</div>
<div class="card-body ">
```

```

<a href="index.PHP?m=ticket&s=ticket_add" class="btn btn-primary">Tambah Ticket</a>

<div class="table-responsive">
<table class="table table-hover">
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Created At</th>
<th>Created By</th>
<th>File </th>
<th>Task to</th>
<th>Deadline</th>
<th>Status</th>
<th>Aksi</th>
</tr>
</thead>
<tbody>
<?PHP
    $data = $db->query("SELECT a.id AS
id_ticket,a.title,a.description,a.created_at,a.created_by,a.File,a.status,a.deadline,a
.id_Users,
(
    SELECT b.nama FROM Users b WHERE a.id_Users=b.id
) AS nama
FROM ticket a ORDER BY a.id DESC", 0);
while($row = $data->fetch_assoc()) {

```

```

?>

<tr>

    <td><?= $row['title'] ?></td>

    <td><?= $row['description'] ?></td>

    <td><?= $row['created_at'] ?></td>

    <td><?= $row['created_by'] ?></td>

    <td>

        <?PHP

        if (!empty($row['File'])) {

            $json = json_decode($row['File']);

            for ($i=0; $i < count($json); $i++) {

                ?>

                <a href="assets/uploads/ticket/<?= $json[$i] ?>" target="_blank">Attachment <?= $i+1 ?></a><br>

                <?PHP } } else { echo '-'; } ?>

            </td>

            <td><?= $row['nama'] ? $row['nama'] : 'Belum di-assign' ?></td>

            <td><?= $row['deadline'] ?></td>

            <td><?= $row['status'] == 0 ? 'Work in Progess' : ($row['status'] == 1 ? 'Pending' : 'Done') ?></td>

            <td>

                <a href="index.PHP?m=ticket&s=ticket_edit&id=<?=$row['id_ticket'] ?>" class="btn btn-xs btn-primary">Edit</a>

                <a onclick="DeleteData(<?= $row['id_ticket'] ?>)" class="btn btn-xs btn-danger" style="color:#fff;cursor:pointer">Hapus</a>

            </td>

        </tr>

```

```
<?PHP } ?>
</tbody>
</table>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<script>
function DeleteData(id) {
    var r = confirm("Anda yakin ingin menghapus ini");
    if (r == true) {
        location.href = "pages/ticket/ticket_proses.PHP?aksi=hapus&id=" + id;
    }
}
</script>
```

5.5.4.2 Perancangan Antarmuka Add Data Ticket

The screenshot shows a user interface for managing tickets. On the left, there's a sidebar with links: HOME, USERS, DEPARTMENT, TICKET (with a blue circular icon), and LOGOUT. The main area is titled 'Edit Ticket' and contains the following fields:

- Assign To:** A dropdown menu labeled 'Select User'.
- Title:** A text input field.
- Description:** A text input field.
- File:** A file input field with a 'Choose File' button and a message 'No File Chosen'.
- Tambah:** A green button.
- Date Input:** A dropdown menu for selecting a date in 'dd----yy' format.
- Submit:** A green button.

Gambar 5. 30 Antarmuka Add Data Ticket

Berikut *Source Code* untuk halaman Add Data Ticket :

```
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card ">
<div class="card-header ">
<h5 class="card-title">Tambah Ticket</h5>
</div>
<div class="card-body ">
<form action="pages/ticket/ticket_proses.PHP?aksi=insert" method="POST"
encType="multipart/form-data">
<div class="form-group">
```

```

<label for="inputText3" class="col-form-label">Assign To</label>
<select class="default-select2 form-control" name="id_Users">
    <option value="">-- Select Users --</option>
    <?PHP
        $Users = $db->query("SELECT id, nama FROM Users WHERE
role != 'Admin' ORDER BY id ASC");

        foreach ($Users as $row_Users) {
    ?>
            <option value="<?= $row_Users['id'] ?>"><?= $row_Users['nama']
?></option>
        <?PHP } ?>
    </select>
</div>

<div class="form-group">
    <label for="inputText3" class="col-form-label">Title</label>
    <input id="inputText3" name="title" Type="Text" class="form-control">
</div>

<div class="form-group">
    <label for="inputText3" class="col-form-label">Description</label>
    <Textarea class="form-control" name="description"
rows="4"></Textarea>
</div>

<div id="image">
    <div id="item">
        <div class="form-group">
            <label for="inputText3" class="col-form-label">File</label>

```

```

<input id="inputText3" name="File[]" Type="File" class="form-control" style="position:unset;opacity:1">
</div>
</div>

<a href="javascript:void(0)" onclick="tambahFile()" class="btn btn-primary">Tambah</a>
</div>
<div class="form-group">
    <label for="inputText3" class="col-form-label">Deadline</label>
    <input id="inputText3" name="deadline" Type="datetime-local" class="form-control">
</div>
<div class="form-group">
    <input Type="submit" class="btn btn-primary" value="Submit">
</div>
</form>
</div>
</div>
</div>
<script>
function tambahFile() {
    var html = '<div id="item"><div class="form-group"><input id="inputText3" name="File[]" Type="File" class="form-control" style="position:unset;opacity:1"></div><a href="javascript:void(0)" onclick="removeFile(this)" class="btn btn-danger">Hapus</a></div>';

```

```

        $('#image').append(html);

    }

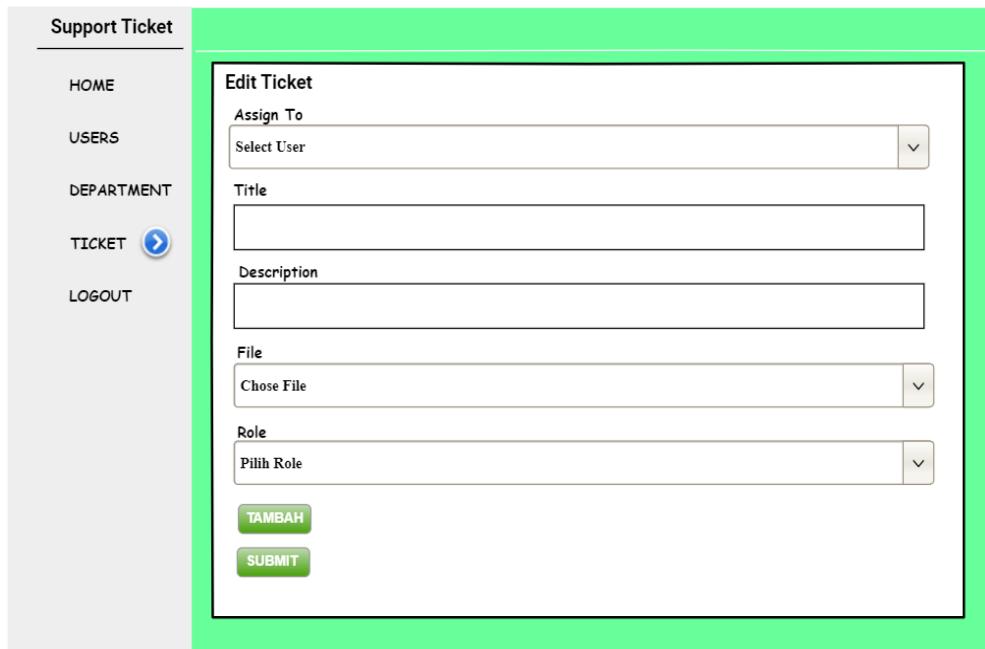
function removeFile(e) {
    let parent = e.parentNode;

    parent.remove()
}

}
</script>

```

5.5.4.3 Perancangan Antarmuka *Edit Data Ticket*



Gambar 5. 31 Antarmuka *Edit Data Ticket*

Berikut *Source Code* untuk halaman *Edit Data Ticket* :

```

<?PHP

    $data = $db->query("SELECT * FROM ticket WHERE id=\"".$_GET['id']."'"
LIMIT 1", 0);

    $row = $data->fetch_assoc();

?>

<div class="content">

<div class="row">

<div class="col-md-12">

<div class="card ">

    <div class="card-header ">

        <h5 class="card-title">Edit Ticket</h5>

    </div>

    <div class="card-body ">

        <form action="pages/ticket/ticket_proses.PHP?aksi=Update&id=<?=
$_GET['id'] ?>" method="POST" encType="multipart/form-data">

            <div class="form-group">

                <label for="inputText3" class="col-form-label">Assign To</label>

                <select class="default-select2 form-control" name="id_Users">

                    <option value="">-- Select Users --</option>

                    <?PHP

                        $Users = $db->query("SELECT id, nama FROM Users WHERE
role != 'Admin' ORDER BY id ASC");



                    foreach ($Users as $row_Users) {

                        ?>

                        <option value="<?= $row_Users['id'] ?><?= $row_Users['nama']
?></option>

```

```

<?PHP } ?>

</select>

</div>

<div class="form-group">

    <label for="inputText3" class="col-form-label">Title</label>

    <input id="inputText3" name="title" Type="Text" class="form-control" value="<?= $row['title'] ?>">

</div>

<div class="form-group">

    <label for="inputText3" class="col-form-label">Description</label>

    <Textarea class="form-control" name="description" rows="4"><?= $row['description'] ?></Textarea>

</div>

<div id="image">

    <div id="item">

        <div class="form-group">

            <label for="inputText3" class="col-form-label">File</label>

            <input id="inputText3" name="File[]" Type="File" class="form-control" style="pointer-events:unset; opacity:1">

        </div>

    </div>

    <a href="javascript:void(0)" onclick="tambahFile()" class="btn btn-primary">Tambah</a>

</div>

<div class="form-group">

    <input Type="submit" class="btn btn-primary" value="Submit">

</div>

```

```
</form>

</div>

</div>

</div>

</div>

<script>

function tambahFile() {

    var html = '<div id="item"><div class="form-group"><input id="inputText3"
name="File[]" Type="File" class="form-control"
style="position:unset;opacity:1"></div><a href="javascript:void(0)"
onclick="removeFile(this)" class="btn btn-danger">Hapus</a></div>';

    $('#image').append(html);

}

function removeFile(e) {

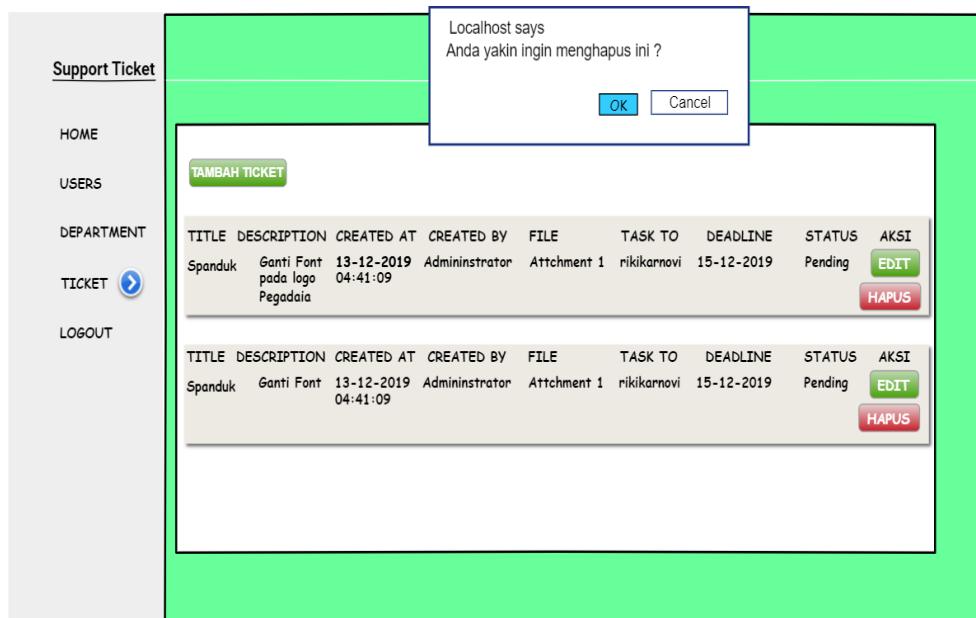
    let parent = e.parentNode;

    parent.remove()

}

</script>
```

5.5.4.4 Perancangan Antarmuka Delete Data Ticket



Gambar 5. 32 Perancangan Antarmuka Delete data Ticket

Berikut *Source Code* untuk halaman Delete data Ticket :

```
<?PHP  
session_start();  
require '../vendor/autoload.PHP';  
use PHPMailer\PHPMailer\PHPMailer;  
  
include '../db/db.PHP';  
$aksi = $_GET['aksi'];  
  
// MEMBUAT FUNCTION SENDING EMAIL  
function sendMail($email, $nama, $title, $message) {  
    $mail = new PHPMailer;
```

```

$mail->isSMTP();
$mail->IsHTML(true);
$mail->Host = 'smtp.gmail.com';
$mail->Port = 587;
$mail->SMTPSecure = 'tls';
$mail->SMTPAuth = true;
$mail->Username = "smtpgratisemail@gmail.com";
$mail->Password = "SMTPGratis-99";

$mail->setFrom('riki-noreply@gmail.com', '[no-reply] Riki');
$mail->addAddress($email, $nama);

$mail->Subject = $title;
$mail->Body = $message;

$mail->send();
}

// MEMBUAT FUNCTION SENDING TELEGRAM
function sendTelegram($chat_id, $message) {
    $apiToken = "992299951:AAHx8fW2xdppwO9WOnn0BFE1EQaVyiVCVa8";

    $data = [
        'chat_id' => '@' . $chat_id,
        'Text' => $message
    ];
}

```

```

try {
    $response =
File_get_contents("https://api.telegram.org/bot".$apiToken."/sendMessage?chat_
id=".$chat_id."&Text=".urlencode($message));

} catch (\Throwable $th) {

}

}

// PROSES CHANGE STATUS TIKET

if ($aksi == 'Change_status') {

$id = (int)$_GET['id'];

$status = (int)$_GET['status'];

$Update = $db->query("UPDATE ticket SET status='".$status."' WHERE
id='".$id."');

if ($Update) {

$Users = $db->query("SELECT email, nama FROM Users WHERE role='Admin'
LIMIT 1", 0);

$row_Users = $Users->fetch_assoc();

$nama = $row_Users['nama'];

$email = $row_Users['email'];

$ticket = $db->query("SELECT id_Users, title FROM ticket WHERE id='".$id."
LIMIT 1", 0);

$row_ticket = $ticket->fetch_assoc();

```

```

$assign = $db->query("SELECT email, nama FROM Users WHERE
id='".$row_ticket['id_Users']."' LIMIT 1", 0);

$row_assign = $assign->fetch_assoc();

$nama_Users = $row_assign['nama'];
$email_Users = $row_assign['email'];

$ticket_title = $row_ticket['title'];

$status_Changed = ($status == 0 ? 'Work in Progress' : ($status == 1 ? 'Pending' :
'Done'));

$message_1 = '<!DOCTYPE html><html lang="en"><head> <meta charset="utf-
8"/> <title>Email Status Changed!</title> <meta content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, Users-scalable=no" name="Viewport"/>
<meta content="" name="description"/> <meta content="" name="author"/></head><body style="font-family: Helvetica, Arial, sans-serif;-
webkit-Text-size-adjust: 100%;-ms-Text-size-adjust: 100%;-webkit-font-smoothing: antialiased;-moz-osx-font-smoothing: grayscale;padding:20px;"> <div style="max-
width: 600px;margin: 0 auto;background-color: #ddd;padding:10px 20px;border-
radius:10px;"> <h4 align="center" style="Text-align:center;margin:20px 0;font-
size:18px;">Email Status Changed!</h4> <p style="Text-align: center;font-size:
16px;letter-spacing: .5px;">Hai, <span style="font-
weight:600">'$.nama.'</span></p><p style="Text-align: center;font-size:
14px;">Email ini berfungsi untuk memberitahu Administrator bahwa status ticket
<b>'$.ticket_title.'</b> sedang dirubah menjadi
<b>'$.status_Changed.'</b>.</p><p style="Text-align: center;margin-top:
20px;"><a href="http://localhost/riki" target="_blank" style="font-size: 16px;font-
weight: 600;letter-spacing: .5px;border-radius:5px;display: inline-
block;background-color: #00acac;color: #fff !important;padding: 10px 15px;Text-
decoration: none;">Cek Aktivitas</a></p></div></body></html>

';

```

```
$message_2 = '<!DOCTYPE html><html lang="en"><head> <meta charset="utf-8"/> <title>Email Status Changed!</title> <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, User-scalable=no" name="viewport"/> <meta content="" name="description"/> <meta content="" name="author"/></head><body style="font-family: Helvetica, Arial, sans-serif;-webkit-Text-size-adjust: 100%;-ms-Text-size-adjust: 100%;-webkit-font-smoothing: antialiased;-moz-osx-font-smoothing: grayscale;padding:20px;"> <div style="max-width: 600px;margin: 0 auto;background-color: #ddd;padding:10px 20px;border-radius:10px;"> <h4 align="center" style="Text-align:center;margin:20px 0;font-size:18px;">Email Status Changed!</h4> <p style="Text-align: center;font-size: 16px;letter-spacing: .5px;">Hai, <span style="font-weight:600">'.$nama_Users.'</span></p><p style="Text-align: center;font-size: 14px;">Email ini berfungsi untuk memberitahu Anda bahwa status ticket <b>'.$ticket_title.'</b> sedang dirubah menjadi <b>'.$status_Changed.'</b>.</p><p style="Text-align: center; margin-top: 20px;"><a href="http://localhost/riki" target="_blank" style="font-size: 16px;font-weight: 600;letter-spacing: .5px; border-radius:5px; display: inline-block; background-color: #00acac; color: #fff !important; padding: 10px 15px; Text-decoration: none;">Cek Aktivitas</a></p></div></body></html>' ;
```

```
$title = 'Email Status Changed!';
```

```
sendMail($email, $nama, $title, $message_1);
```

```
sendMail($email_Users, $nama_Users, $title, $message_2);
```

```
// $message_tele = 'Hai, '.$nama_Users.'. Email ini berfungsi untuk memberitahu Anda bahwa status ticket '.$ticket_title.' sedang dirubah menjadi '.$status_Changed; // sendTelegram('892716190', $message_tele);
```

```
echo 1;
```

```
} else {
```

```

echo 0;

}

// MELIHAT DETAIL TIKET YABF DIBUAT

} else if ($aksi == 'get_detail') {

    $id = (int)$_GET['id'];

    $query = $db->query("SELECT * FROM ticket WHERE id=\"$id\" LIMIT 1", 0);
    $row = $query->fetch_assoc();

    $attach = "";

    try {
        $json = json_decode($row['File']);
        $count = count($json);

        if ($count > 0) {
            for ($i=0; $i < count($json); $i++) {
                $attach .= '<a href="assets/uploads/ticket/'.$json[$i].'"'
                target="_blank">Attachment ' .($i+1). '</a><br>';
            }
        } else {
            $attach = '-';
        }
    } catch (Exception $e) {
        $attach = '-';
    }
}

```

```

$activity_html = "";

$activity = $db->query("SELECT * FROM ticket_activity WHERE
ticket_id='".$id."'", 0);

while ($row_activity = $activity->fetch_assoc()) {

    $activity_html .= '

<div style="margin: 5px 0;">

    <div style="border:1px solid #ddd; border-radius:
5px; padding:10px;">

        <div style="font-size: 14px; font-weight:
600;">' . $row_activity['activity_from'] . '</div>

        <div style="font-size: 12px; margin: 10px
0;">' . $row_activity['message'] . '</div>

        <div style="font-size: 10px;">' . $helpers-
>dateTimeIndonesia($row_activity['created_at']) . '</div>

    </div>

</div>

';

}

$html = '

<div class="modal-header">

<h6 class="modal-title" id="label">Title : ' . $row['title'] . '</h6>

<button Type="button" class="close" data-dismiss="modal" aria-
label="Close">

    <span aria-hidden="true">&times;</span>

</button>

</div>

```

```

<div class="modal-body">
    <h6>Description</h6>
    <div style="margin-bottom: 20px;font-size:12px;">' . $row['description'] .
'</div>

    <h6>Attachment</h6>
    <div style="margin-bottom: 20px;font-size:12px;">' . $attach . '</div>

    <h6>Activity</h6>
    <input Type="hidden" id="id_ticket" value="' . $row['id'] . "' />
    <Textarea id="activity" class="form-control" style="margin-
bottom:10px;padding:5px 10px;" placeholder="Tuliskan activity
disini"></Textarea>

    <a href="javascript:void(0)" onclick="submitActivity()" class="btn btn-block
btn-secondary">SUBMIT</a>

    <div style="max-height:300px;overflow-y:auto;overflow-
x:hidden;border:1px solid #ddd;padding:5px;">' . $activity_html . '</div>

</div>

<div class="modal-footer">
    <button Type="button" class="btn btn-danger" data-
dismiss="modal">Close</button>
</div>';

echo $html;

// MEMBUAT AKTIVITAS TIKET KEPADA USERS

} else if ($aksi == 'submit_activity') {

    $data = json_decode(File_get_contents('PHP://input'), true);

    $id = $data['id'];
    $activity = $data['activity'];

```

```

$created_at = date('Y-m-d H:i:s');

$activity_from = $_SESSION['nama'];

$insert = $db->query('INSERT INTO ticket_activity (
    message,
    ticket_id,
    activity_from,
    created_at
) VALUES (
    "'.$activity.'",
    "'.$id.'",
    "'.$activity_from.'",
    "'.$created_at.'"
)');
echo 1;

// INPUT DATA TIKET KEDALAM DATABASE

} else if ($aksi == 'insert') {

$title = $_POST['title'];

$description = $_POST['description'];

$created_at = date('Y-m-d H:i:s');

$created_by = $_SESSION['nama'];

$id_Users = $_POST['id_Users'];

$deadline = $_POST['deadline'];

if (!empty($_FILES['File']['name'][0])) {

```

```

$count = count($_FILES['File']['name']);

$Files = [];

for ($i=0; $i < $count; $i++) {
    $File_name = $_FILES['File']['name'][$i];
    $File_tmp =$_FILES['File']['tmp_name'][$i];
    $File_Type = $_FILES['File']['Type'][$i];
    $File_ext = end(explode(".", $_FILES['File']['name'][$i]));

    $File_upload = 'ticket-' . date('YmdHis') . '.' . $File_ext;

    array_push($Files, $File_upload);

    move_uploaded_file($File_tmp,
    "../../../../assets/uploads/ticket/".$File_upload);
}

$File_save = json_encode($Files);

$insert = $db->query("INSERT INTO ticket (
    File,
    title,
    description,
    created_at,
    created_by,
    id_Users,
    deadline
");

```

```

) VALUES (
    "'.$File_save.'",
    "'.$title.'",
    "'.$description.'",
    "'.$created_at.'",
    "'.$created_by.'",
    "'.$id_Users.'",
    "'.$deadline.'"
);

} else {
    $insert = $db->query('INSERT INTO ticket (
        title,
        description,
        created_at,
        created_by,
        id_Users,
        deadline
    ) VALUES (
        "'.$title.'",
        "'.$description.'",
        "'.$created_at.'",
        "'.$created_by.'",
        "'.$id_Users.'",
        "'.$deadline.'"
    );
}

```

```

if ($insert) {

    $assign = $db->query("SELECT email, nama FROM Users WHERE
id='".$id_Users."' LIMIT 1", 0);

    $row_assign = $assign->fetch_assoc();

    $nama = $row_assign['nama'];
    $email = $row_assign['email'];

    $ticket_title = $title;

    $message = '<!DOCTYPE html><html lang="en"><head> <meta
charset="utf-8"/> <title>You have new ticket!</title> <meta
content="width=device-width, initial-scale=1.0, maximum-scale=1.0, Users-
scalable=no" name="Viewport"/> <meta content="" name="description"/> <meta
content="" name="author"/></head><body style="font-family: Helvetica, Arial,
sans-serif;-webkit-Text-size-adjust: 100%;-ms-Text-size-adjust: 100%;-webkit-font-
smoothing: antialiased;-moz-osx-font-smoothing: grayscale;padding:20px;"> <div
style="max-width: 600px;margin: 0 auto;background-color: #ddd;padding:10px
20px;border-radius:10px;"> <h4 align="center" style="Text-
align:center;margin:20px 0;font-size:18px;">You have new ticket!</h4> <p
style="Text-align: center;font-size: 16px;letter-spacing: .5px;">Hai, <span
style="font-weight:600">'$nama.'</span></p><p style="Text-align: center;font-
size: 14px;">Email ini berfungsi untuk memberitahu Anda bahwa anda memiliki
ticket baru dengan title <b>'$ticket_title.'</b>, silahkan cek ticket anda dengan
menekan tombol dibawah.</p><p style="Text-align: center;margin-top: 20px;"><a
href="http://localhost/riki" target="_blank" style="font-size: 16px;font-weight:
600;letter-spacing: .5px;border-radius:5px;display: inline-block;background-color:
#00acac;color: #fff !important;padding: 10px 15px;Text-decoration: none;">Cek
Aktivitas</a></p></div></body></html>

';

}

$title = 'You have new ticket!';

```

```

sendMail($email, $nama, $title, $message);

echo '<script>alert("Data berhasil ditambahkan");location.href =
"../../index.PHP?m=ticket&s=ticket"</script>';

} else {

echo '<script>alert("Data gagal ditambahkan");history.go(-1)</script>';

}

// PROSES UPDATE DATA TIKET

} else if ($aksi == 'Update') {

$id = $_GET['id'];

$title = $_POST['title'];

$description = $_POST['description'];

$id_Users = $_POST['id_Users'];

if (!empty($_FILES['File']['name'][0])) {

$count = count($_FILES['File']['name']);

$Files = [];

for ($i=0; $i < $count; $i++) {

$File_name = $_FILES['File']['name'][$i];

$File_tmp =$_FILES['File']['tmp_name'][$i];

$File_Type = $_FILES['File']['Type'][$i];

$File_ext = end(explode(".", $_FILES['File']['name'][$i]));

$File_upload = 'ticket-' . date('YmdHis') . '.' . $File_ext;

array push($Files, $File_upload);

}
}
}

```

```

        move_uploaded_File($File_tmp,
"../../assets/uploads/ticket/".$File_upload);

    }

$File_save = json_encode($Files);

$Update = $db->query("UPDATE ticket SET
File='".$File_save."',
title='".$title."',
id_Users='".$id_Users."',
description='".$description."'"

WHERE id='".$id."');

} else {

$Update = $db->query("UPDATE ticket SET
title='".$title."',
id_Users='".$id_Users."',
description='".$description."'"

WHERE id='".$id."');

}

if ($Update) {

$assign = $db->query("SELECT email, nama FROM Users WHERE
id='".$id_Users."' LIMIT 1", 0);

$row_assign = $assign->fetch_assoc();

$nama = $row_assign['nama'];

```

```
$email = $row_assign['email'];

$ticket_title = $title;

$message = '<!DOCTYPE html><html lang="en"><head> <meta charset="utf-8"/> <title>You have new ticket!</title> <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, Users-scalable=no" name="Viewport"/> <meta content="" name="description"/> <meta content="" name="author"/></head><body style="font-family: Helvetica, Arial, sans-serif;-webkit-Text-size-adjust: 100%;-ms-Text-size-adjust: 100%;-webkit-font-smoothing: antialiased;-moz-osx-font-smoothing: grayscale;padding:20px;"> <div style="max-width: 600px;margin: 0 auto;background-color: #ddd;padding:10px 20px;border-radius:10px;"> <h4 align="center" style="Text-align:center;margin:20px 0;font-size:18px;">You have new ticket!</h4> <p style="Text-align: center;font-size: 16px;letter-spacing: .5px;">Hai, <span style="font-weight:600">'.$nama.'</span></p><p style="Text-align: center;font-size: 14px;">Email ini berfungsi untuk memberitahu Anda bahwa anda memiliki ticket baru dengan title <b>' . $ticket_title . '</b>, silahkan cek ticket anda dengan menekan tombol dibawah.</p><p style="Text-align: center; margin-top: 20px;"><a href="http://localhost/riki" target="_blank" style="font-size: 16px;font-weight: 600;letter-spacing: .5px; border-radius:5px; display: inline-block; background-color: #00acac; color: #fff !important; padding: 10px 15px; Text-decoration: none;">Cek Aktivitas</a></p></div></body></html>

';


```

```
$title = 'You have new ticket!';
```

```
sendMail($email, $nama, $title, $message);

echo '<script>alert("Data berhasil diedit");location.href =
"../../index.PHP?m=ticket&s=ticket"</script>';

} else {

echo '<script>alert("Data gagal diedit");history.go(-1)</script>';

}
```

```

// PROSES HAPUS DARA TIKET
} else if ($aksi == 'hapus') {
    $id = $_GET['id'];
    $hapus = $db->query("DELETE FROM ticket WHERE id='".$id."'");
}

if ($hapus) {
    echo '<script>alert("Data berhasil dihapus");location.href =
"../../index.PHP?m=ticket&s=ticket"</script>';
} else {
    echo '<script>alert("Data gagal dihapus");history.go(-1)</script>';
}
}
}

```

5.5.5 Perancangan Antarmuka *Department*

5.5.5.1 Perancangan Antarmuka *View Data Department*



Gambar 5. 33 Antarmuka *View Data Department*

Berikut *Source Code* untuk halaman *View Data Department* :

```
<div class="content">

<div class="row">
    <div class="col-md-12">
        <div class="card ">
            <div class="card-header ">
                <h5 class="card-title">Department</h5>
            </div>
            <div class="card-body ">
                <a href="index.PHP?m=Department&s=Department_add" class="btn btn-primary">Tambah Department</a>
                <div class="table-responsive">
                    <table class="table table-hover">
                        <thead>
                            <tr>
                                <th>No</th>
                                <th>Department</th>
                                <th>Total Users</th>
                                <th>Aksi</th>
                            </tr>
                        </thead>
                        <tbody>
                            <?PHP
                                $data = $db->query("SELECT a.id_dept, a.Department,
                                b.id, b.Department, b.Usersname, b.nama, b.email,
                                b.role,
```

```

        COUNT(b.Department) AS total

        FROM Department a JOIN Users b

        ON a.Department=b.Department GROUP BY

a.Department, b.Department

        ORDER BY id_dept DESC", 0);

$no=0;

while($row = $data->fetch_assoc()) {

$no++;

?>

<tr>

<td><?= $no ?>.</td>

<td><?= $row['Department'] ?></td>

<td><?= $row['total'] ?></td>

<td>

<a

href="index.PHP?m=Department&s=Department_edit&id_dept=<?=
$row['id_dept'] ?>" class="btn btn-xs btn-primary">Edit</a>

<a onclick="DeleteData(<?= $row['id_dept'] ?>)" class="btn btn-xs
btn-danger" style="color:#fff;cursor:pointer">Hapus</a>

</td>

</tr>

<?PHP } ?>

</tbody>

</table>

</div>

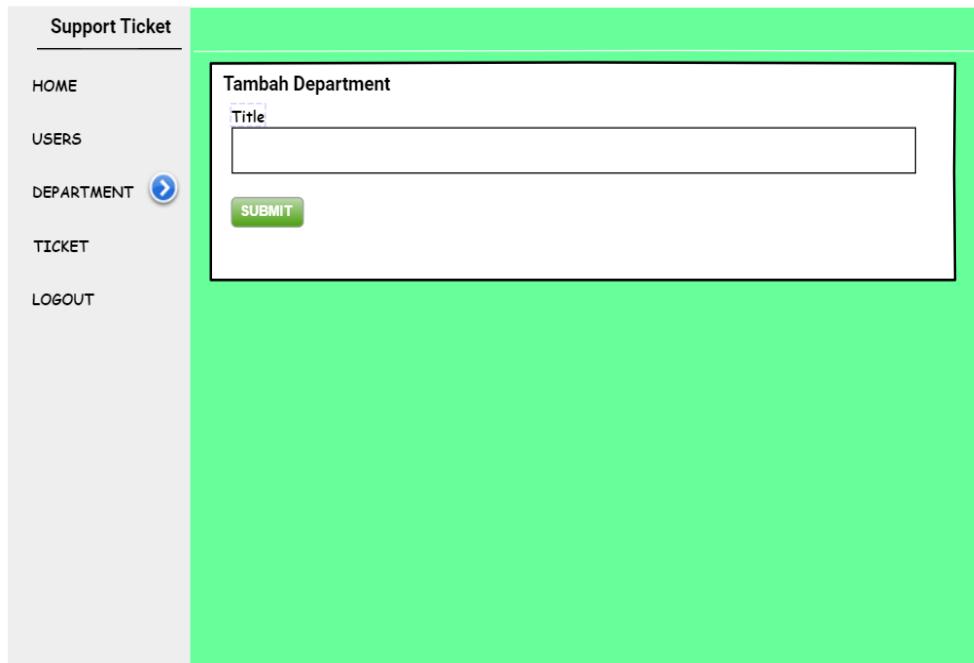
</div>

</div>

```

```
</div>
</div>
</div>
<script>
function DeleteData(id_dept) {
    var r = confirm("Anda yakin ingin menghapus ini");
    if (r == true) {
        location.href =
"pages/Department/Department_proses.PHP?aksi=hapus&id_dept=" + id_dept;
    }
}
</script>
```

5.5.5.2 Perancangan Antarmuka Add Data Department



Gambar 5. 34 Antarmuka Add Data Department

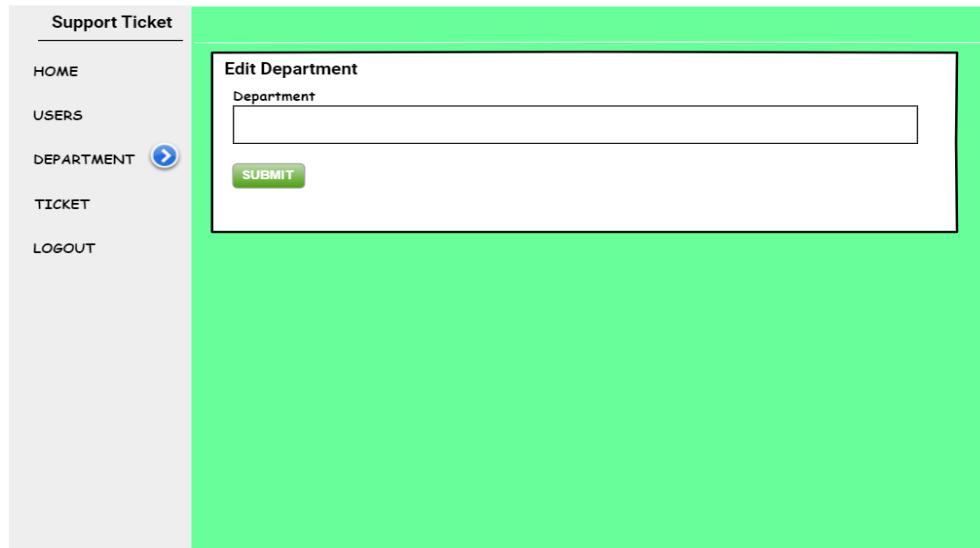
Berikut *Source Code* untuk halaman *Add Data Department* :

```
<div class="content">
<div class="row">
<div class="col-md-12">
<div class="card ">
<div class="card-header ">
<h5 class="card-title">Tambah Department</h5>
</div>
<div class="card-body ">
<form action="pages/Department/Department_proses.PHP?aksi=insert"
method="POST">
<div class="form-group">
```

```
<label for="inputText3" class="col-form-label">Title</label>
<input id="inputText3" name="Department" Type="Text" class="form-control">
</div>
<div class="form-group">
<input Type="submit" class="btn btn-primary" value="Submit">
</div>
</form>
</div>
</div>
</div>
</div>
<script>
function removeFile(e) {
    let parent = e.parentNode;

    parent.remove()
}
</script>
```

5.5.5.3 Perancangan Antarmuka *Edit Data Department*



Gambar 5. 35 Antarmuka *Edit Data Department*

Berikut *Source Code* untuk halaman *Edit Data Department* :

```
<?PHP  
  
$data = $db->query('SELECT * FROM Department WHERE  
id_dept="'. $_GET['id_dept']. '"');  
  
$row = $data->fetch_assoc()  
  
?>  
  
<div class="content">  
  <div class="row">  
    <div class="col-md-12">  
      <div class="card ">  
        <div class="card-header ">  
          <h5 class="card-title">Edit Department</h5>  
        </div>  
        <div class="card-body ">
```

```
<form  
action="pages/Department/Department_proses.PHP?aksi=Update&id_dept=<?=  
$_GET['id_dept'] ?>" method="POST">  
  
    <div class="form-group">  
        <label for="inputText3" class="col-form-label">Department</label>  
        <input id="inputText3" name="Department" Type="Text" value="<?= $row['Department'] ?>" class="form-control"/>  
    </div>  
  
    <div class="form-group">  
        <input Type="submit" class="btn btn-primary" value="Submit">  
    </div>  
    </div>  
    </div>  
    </div>  
    </div>  
    </div>
```

5.5.5.4 Perancangan Antarmuka Delete Data Department



Gambar 5. 36 Antarmuka *Delete Data Department*

Berikut *Source Code* untuk halaman *Edit Data Department* :

```
<?PHP  
include '../db/db.PHP';  
$aksi = $_GET['aksi'];  
  
// INPUT DATA DEPARTMENT  
if ($aksi == 'insert') {  
    $Department = $_POST['Department'];
```

```

$insert = $db->query('INSERT INTO Department (Department) VALUES
("'.$Department.'")');

if ($insert) {
    echo '<script>alert("Data berhasil ditambahkan");location.href =
"../../index.PHP?m=Department&s=Department"</script>';
} else {
    echo '<script>alert("Data gagal ditambahkan");history.go(-1)</script>';
}

// UPDATE DATA DEPARTMENT

} else if ($aksi == 'Update') {
    $id_dept = $_GET['id_dept'];
    $Department = $_POST['Department'];

    $Update = $db->query('UPDATE Department SET Department="'. $Department .'"'
WHERE id_dept="'. $id_dept .'");

    if ($Update) {
        echo '<script>alert("Data berhasil diedit");location.href =
"../../index.PHP?m=Department&s=Department"</script>';
    } else {
        echo '<script>alert("Data gagal diedit");history.go(-1)</script>';
    }
}

// HAPUS DATA DEPARTMENT

} else if ($aksi == 'hapus') {
    $id_dept = $_GET['id_dept'];
    $hapus = $db->query('DELETE FROM Department WHERE
id_dept="'. $id_dept .'");

```

```
if ($hapus) {  
    echo '<script>alert("Data berhasil dihapus");location.href =  
"../../index.PHP?m=Department&s=Department"</script>';  
}  
} else {  
    echo '<script>alert("Data gagal dihapus");history.go(-1)</script>';  
}  
}
```

BAB VI

KESIMPULAN

6.1 Kesimpulan

Setelah melakukan analisis, perancangan dan pembangunan aplikasi Sistem *Monitoring Job Desk Human Capital* menggunakan *PHP Native*, maka dapat disimpulkan bahwa aplikasi yang dibangun telah mampu menjawab permasalahan yang di bahasa *Source Code* untuk halaman *Login*, serta telah berhasil mencapai tujuan yaitu :

1. Membangun aplikasi *Monitoring* pada divisi *Human Capital* dengan menggunakan *PHP Native*, yang diberi nama Sistem *Monitoring Job Desk Human Capital*.
2. Pemberian tugas langsung pada staf yang dituju dan langsung terhubung dengan sistem guna membantu pihak terkait dalam proses pekerjaan yang diberikan.
3. Membantu divisi *human capital* dalam melakukan manajemen pekerjaan yang akan diberikan tanpa perlu melakukan kontak langsung ke pihakt terkait yang akan diberikan pekerjaan tertentu.