MEMBANGUN RANCANGAN APLIKASI *REPOSITORY* SKRIPSI MAHASISWA DEPARTEMEN MATEMATIKA UNIVERSITAS HASANUDDIN BERBASIS *WEB*

SKRIPSI

MIR ATAINI APRILIA

H071171520

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

2022

MEMBANGUN RANCANGAN APLIKASI *REPOSITORY* SKRIPSI MAHASISWA DEPARTEMEN MATEMATIKA UNIVERSITAS HASANUDDIN BERBASIS *WEB*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains pada program studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas hasanuddin

MIR ATAINI APRILIA

H071171520

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

PERNYATAAN KEASLIAN

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini :

Nama : Mir Ataini Aprilia

NIM : H071171520

Program Studi : Sistem Informasi

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul :

Membangun Rancangan Aplikasi Repository Skripsi Mahasiswa Departemen Matematika Universitas Hasanuddin Berbasis Web

Adalah karya tulisan saya sendiri, bukan merupakan pengambil alihan tulisan orang lain dan bahwa skripsi/tesis/disertasi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi/tesis/disertasi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 19 Juli 2022

Mir Ataini Aprilia

ii

LEMBAR PENGESAHAN TUGAS AKHIR

LEMBAR PENGESAHAN TUGAS AKHIR

MEMBANGUN RANCANGAN REPOSITORY SKRIPSI MAHASISWA DEPARTEMEN MATEMATIKA UNIVERSITAS HASANUDDIN BERBASIS WEB

Disusun dan diajukan oleh

MIR ATAINI APRILIA
H071171520

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian studi Program Sarjana Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Pada tanggal 19 Juli 2022

dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui

Pembimbing Utama	Pembimbing Pendamping
Dr. Hendra, S.Si, M.Kom.	**Supri Bin Hj Amir, S.Si., M.Eng.**
NIP. 197601022002121001	**NIP. 198805042019031012**

Ketua Program Studi

Dr. Muhammad Hasbi, M.Sc
NIP. 196307201989031003

HALAMAN PENGESAHAN

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	: Mir Ataini Aprilia
NIM	: H071171520
Program Studi	: Sistem Informasi
Judul Skripsi	:Membangun Rancangan Aplikasi Repository Skripsi Mahasiswa Departemen Matematika Universitas Hasanuddin Berbasis Web

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

		Tanda tangan
Ketua	: Dr. Hendra, S.Si., M.Kom.	(.....)
Sekretaris	: Supri Bin Hj. Amir, S.Si., M.Eng.	(.....)
Anggota	: Rozalina Amran, S.T., M.Eng.	(.....)
Anggota	: Edy Saputra, S.Si., M.Si.	(.....)

Ditetapkan di : Makassar
Tanggal : 19 Juli 2022

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT, karena berkat rahmat dan karunia-Nyalah penulis dapat menyelesaikan skripsi yang berjudul "Membangun Rancangan Aplikasi Repository Skripsi Mahasiswa Departemen Matematika Universitas Hasanuddin Berbasis Web". Dalam skripsi ini dibahas mengenai pengarsipan skripsi yang terkomputerisasi. Adapun maksud dan tujuan dari penulisan skripsi ini adalah untuk memenuhi salah satu syarat untuk mengikuti sidang skripsi, Jurusan Matematika Prodi Sistem Informasi.

Selama penelitian dan penulisan skripsi ini banyak sekali hambatan yang penulis alami, namun berkat bantuan, dorongan serta bimbingan dari berbagai pihak, akhirnya skripsi ini dapat terselesaikan dengan baik. Penulis beranggapan bahwa skripsi ini merupakan karya terbaik yang dapat penulis persembahkan. Tetapi penulis menyadari bahwa tidak tertutup kemungkinan didalamnya terdapat kekurangan-kekurangan. Oleh karena itu kritik dan saran yang membangun sangat penulis harapkan. Akhir kata, semoga skripsi ini dapat bermanfaat bagi penulis khususnya dan bagi para pembaca pada umumnya.

Disamping itu, penulis juga ingin mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada semua pihak yang telah membantu dalam proses penulisan skripsi ini diantaranya adalah:

1. Rektor Universitas Hasanuddin, Bapak Prof. Dr. Ir. Jamaluddin Jompa, M.Sc. beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Dr. Eng. Amiruddin beserta jajarannya.
3. Ketua Departemen Matematika FMIPA, Prof. Dr. Nurdin, S.Si., M.Si, dan juga Drs. Muhammad Hasbi, M.Sc sebagai ketua Program Studi Sistem Informasi Universitas Hasanuddin.
4. Bapak Dr. Hendra, S.Si, M.Kom. sebagai pembimbing utama yang telah banyak memberikan arahan, ide, motivasi kepada penulis dalam banyak hal.
5. Bapak Supri Bin Hj.Amir, S.Si., M.Eng sebagai pembimbing pertama yang telah memberikan masukan dan arahan kepada penulis.
6. Bapak Edy Saputra R, S.Si., M.Si dan ibu Rozalina Amran, S.T., M.Eng. . sebagai tim penguji atas saran dan masukan pada penelitian yang telah dilakukan oleh penulis.

7. Bapak/Ibu Dosen Departemen Matematika Unhas yang telah memberikan ilmu kepada penulis selama menjadi mahasiswa di Departemen Matematika, dan seluruh Staff departemen Matematika dan Ilmu Komputer Unhas yang telah membantu penulis dalam urusan berkas administrasi.
8. Kepada bapak/ibu Badan Penanggulangan Bencana Daerah (BPBD) Sulawesi Selatan yang telah membantu dalam memberikan data-data yang sangat di perlukan selama penulis melaksanakan penelitian.
9. Teruntuk Keluarga, Almh. Mama (Asnarni), Bapak saya tercinta Pak Sofyan, kakak saya Mutmainna, dan Fitriyah yang selalu mendukung, memberi semangat, dukungan moril, nasehat, doa, dan bisa memberikan kekuatan bagi penulis sehingga berada dititik ini.
10. Teruntuk sahabat tercinta Cici, Muthia, Firda, Zarah, Siti, Geby, Eka Fitriani, Eka Kurnia, Dhila yang telah menemani dan memberi bantuan penulis selama perkuliahan.
11. Teman-teman Program Studi Ilmu Komputer 2017 yang telah berjuang bersama dalam suka dan duka selama ini.

Penulis berharap semoga Tuhan Yang Maha Esa mengkaruniakan rahmat dan hidayah-Nya kepada mereka semua. Akhir kata, penulis berterima kasih atas segala bantuan, doa, dan dukungan yang diberikan kepada penulis dan berharap agar Allah SWT membalas segala kebaikan pihak yang telah membantu. Semoga tulisan ini dapat memberikan manfaat bagi para mahasiswa, khususnya bagi Mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam jurusan Matematika dan bagi Perguruan Tinggi.

Makassar, 14 Juli 2022

Mir Ataini Aprilia

ABSTRAK

Aplikasi berbasis web adalah suatu aplikasi yang diakses menggunakan penjelajah web melalui suatu jaringan seperti Internet. Aplikasi web merupakan suatu aplikasi perangkat lunak komputer yang dikodekan dalam bahasa yang didukung penjelajah web (seperti HTML, JavaScript, AJAX, Java, dll) dan bergantung pada penjelajah tersebut untuk menampilkan aplikasi. Aplikasi web menjadi populer karena kemudahan tersedianya aplikasi klien untuk mengaksesnya, penjelajah web, yang kadang disebut sebagai suatu thin client (klien tipis). Aplikasi web yang umum misalnya webmail, toko ritel daring, lelang daring, wiki, papan diskusi, weblog, serta MMORPG. Penelitian ini membahas merancang dan membuat sebuah wadah berbasis Web yang berguna untuk menampung hasil karya mahasiswa berupa karya Ilmiah/Skripsi di Departemen Matematika di Universitas Hasanuddin. Dari hasil penelitian dan pengujian, maka diperoleh hasilberupa sebuah aplikasi repository yang dapat mempermudah mahasiswa/mahasiswi untuk mencari referensiskripsi dengan cepat. Sistem ini memiliki 2 jenis pengguna yaitu admin dan mahasiswa yang masing-masing memiliki peran.

Kata Kunci : HTML, Repository, Webmail, Weblog

ABSTRACT

A web based application is an application that is accessed using a web browser over a network such as the Internet. A web application is a computer software application that is coded in a language the web browser supports (such as HTML, JavaScript, AJAX, Java, etc.) and relies on that browser to display the application. Web applications became popular because of the easy availability of client applications to access them, web browsers, which are sometimes referred to as thin clients. Common web applications such as webmail, online retail stores, online auctions, wikis, discussion boards, weblogs, and MMORPGs. This study discusses designing and creating a Web-based platform that is useful for accommodating student work in the form of Scientific/Thesis works in the Department of Education Mathematics at Hasanuddin University. From the results of research and testing, the results obtained in the form of a repository application that can make it easier for students to find thesis references quickly. This system has 2 types of users, namely admin and students, each of which has a role.

Keywords: HTML, Repository, Webmail, Weblog

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aplikasi berbasis *web* adalah suatu aplikasi yang diakses menggunakan penjelajah *web* melalui suatu jaringan seperti Internet. Aplikasi web juga merupakan suatu aplikasi perangkat lunak komputer yang dikodekan dalam bahasa yang didukung penjelajah *web* (seperti *HTML, JavaScript, AJAX, Java*, dll) dan bergantung pada penjelajah tersebut untuk menampilkan aplikasi. Aplikasi *web* menjadi populer karena kemudahan tersedianya aplikasi klien untuk mengaksesnya, penjelajah *web*, yang kadang disebut sebagai suatu *thin client* (klien tipis). Kemampuan untuk memperbarui dan memelihara aplikasi *web* tanpa harus mendistribusikan dan menginstalasi perangkat lunak pada kemungkinan ribuan komputer klien merupakan alasan kunci popularitasnya. Aplikasi *web* yang umum misalnya webmail, toko ritel daring, lelang daring, wiki, papan diskusi, *weblog*, serta *MMORPG*.

Mahasiswa yang telah selesai melaksanakan ujian akhir harus menyerahkan hasilnya dalam bentuk skripsi yang telah dijilid (*hardcopy*) dan juga CD yang berisikan file-file skripsi dan *source code* program yang dibuat (*softcopy*) kepada staf Departemen Matematika untuk didata dan diarsipkan.

Proses tersebut masih dilakukan dengan cara konvensional, baik itu skripsi yang hanya disimpan dalam lemari arsip maupun pendataannya. Selain itu, apabila ada mahasiswa yang membutuhkan skripsi yang sudah ada, maka mereka harus melakukan pencarian satu persatu pada arsip Departemen.

Sistem pengarispan yang ada pada Departemen Matematika masih dilakukan secara manual untuk mencari informasi tentang Skripsi yang dibuat oleh mahasiswa yang sudah menyelesaikan skripsinya. Sistem secara manual ini mengakibatkan file media yang sudah diberikan kepada admin dapat hilang.

Diharapkan dari hasil penelitian sistem informasi *repository* skripsi ini dengan sistem berbasis komputer yang akan membantu memudahkan bagi mahasiswa tingkat akhir untuk menyelesaikan skripsinya.

1.2 Rumusan Masalah

Sesuai dengan latar belakang di atas, maka dapat ditentukan rumusan masalah untuk penelitian skripsi ini adalah “Bagaimana merancang dan membangun aplikasi *repository* skripsi mahasiswa Departemen Matematika di Universitas Hasanuddin“

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. *Web* ini hanya digunakan untuk mahasiswa Departemen Matematika.
2. *Web* ini dibuat menggunakan *PHP* dan *MySQL* sebagai *database* servernya.
3. Metode yang digunakan pada *web* ini adalah *Waterfall.*
4. Pengujian aplikasi ini menggunakan uji *black box testing*

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk merancang dan membuat sebuah wadah berbasis *Web* yang berguna untuk menampung hasil karya mahasiswa berupa karya Ilmiah/Skripsi di Departemen Matematika di Universitas Hasanuddin

1.5 Manfaat Penelitian

Hasil dari pembuatan tugas akhir ini diharapkan dapat bermanfaat untuk mahasiswa yaitu mempermudah bagi mahasiswa Departemen Matematika untuk mendapatkan informasi berupa karya ilmiah/Skripsi.

BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Sistem

Sistem adalah sekumpulan subsistem, komponen ataupun *element* yang saling bekerja sama dengan tujuan yang sama untuk menghasilkan *output* yang sudah ditentukan sebelumnya (Mulyani, 2016).

Sistem adalah sekumpulan dari beberapa elemen (yang berinteraksi) agar memiliki sebuah pencapaian dan tujuan tertentu (Jogianto, 2005).

Sistem adalah suatu kumpulan atau himpunan dari suatu unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu (Sutabri, 2012).

Sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi dan tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses tertentu (Fathansyah, 2015).

Berdasarkan pengertian di atas penulis menyimpulkan sistem adalah sekumpulan prosedur atau tahapan yang saling berhubungan untuk mencapai sebuah tujuan tertentu. Kata sistem banyak sekali digunakan dalam percakapan sehari-hari, dalam forum diskusi maupun dokumen ilmiah. Kata ini digunakan untuk banyak hal, dan pada banyak bidang pula, sehingga maknanya menjadi beragam. Dalam pengertian yang paling umum, sebuah sistem adalah sekumpulan benda yang memiliki hubungan di mereka.

2.2 Karakter Sistem

Suatu sistem mempunyai ciri-ciri karakteristik yang terdapat pada sekumpulan elemen yang harus dipahami dalam megidentifikasi pembuatan sistem. Adapun karakteristik sistem (Sutabri, 2012) yang dimaksud adalah sebagai berikut:

1. Komponen Sistem

   Suatu *system* terdiri dari sejumlah komponen yang saling berinteraksi dan bekerja sama untuk membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut dengan Supra sistem.

2. Batasan sistem (*boundary*)

   Daerah yang membatasi antara sistem dengan sistem lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan luar sistem (*environment*)

   Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut, yang dengan demikian lingkungan luar tersebut harus selalu dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung sistem (*interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukkan sistem (*input*)

   Energi yang dimasukkan kedalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Sebagai contoh, didalam suatu unit sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputer. Sementara data adalah sinyal *input* yang akan diolah menjadi informasi

6. Keluaran sistem (*output*)

   Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Seperti contoh sistem informasi, keluaran yang dihasilkan adalah informasi, di mana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang merupakan input bagi subsistem lainnya.

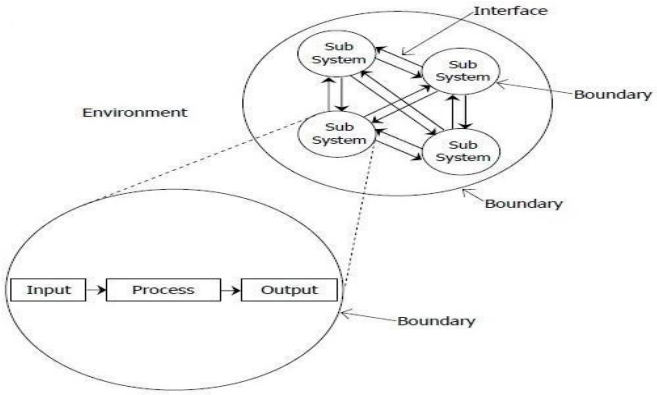
7. Pengolah sistem

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran sistem

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

Inilah gambaran Karakteristik Sistem sebagai berikut :



Gambar 2.1 Karakteristik Sistem

Sumber: (Sutabri, 2012)

2.3 Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dan komponen lain karena sistem memiliki sasaran yang berbeda untuk setiap kasus yang terjadi di dalam sistem tersebut. Oleh karena itu sistem dapat diklasifikasikan dari beberapa sudut pandang. Adapun klasifikasi sistem menurut (Hutahaean, 2015) diuraikan sebagai berikut:

1. Sistem Abstrak dan Sistem Fisik

   Sistem abstrak merupakan sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem telogi. Sedangkan sistem fisik diartikan sebagai sistem yang nampak secara fisik sehingga setiap mahluk dapat melihatnya, misalnya sistem komputer.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem alamiah merupakan sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, misalnya sistem tatas surya, sistem galaksi, sistem reproduksi dan lain-lain. Sedangkan sistem buatan manusia merupakan sistem yang dirancang

oleh manusia. Sistem buatan yang melibatkan interaksi manusia, misalnya sistem akuntansi, sistem informasi, dan lain-lain.

3. Sistem Deterministik dan Sistem Probabilistik

   Sistem deterministik merupakan sistem yang beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi bagian-bagiannya dapat dideteksi dengan pasti sehingga keluaran dari sistem dapat diramalkan, misalnya sistem komputer, adalah contoh sistem yang tingkah lakunya dapat dipastikan berdasarkan program-program komputer yang dijalankan. Sedangkan sistem robabilistik merupakan sistem yang kondisi masa depanya tidak dapat diprediksi karena mengandung unsur probabilitas, misalnya sistem manusia.

4. Sistem Terbuka dan Sistem Tertutup

   Sistem Terbuka merupakan sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Lebih sepesifik dikenal juga yang disebut dengan sistem terotomasi, yang merupakan bagian dari sistem buatan manusia dan beriteraksi dengan kontrol oleh satu atau lebih komputer sebagai bagian dari sistem yang digunakan dalam masyarakat modern. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainnya, misalnya sistem kebudayaan manusia. Sedangkan sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa danya campur tangan dari pihak luar. Secara teoritis sistem tersebut ada, tetapi kenyataannya tidak ada sistem yang benarbenar tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup).

2.4 Elemen Pada Sistem

Ada beberapa elemen yang membentuk sebuah sistem, yaitu : tujuan, masukan, proses, keluaran, batas, mekanisme pengendalian dan umpan balik serta lingkungan. Berikut penjelasan mengenai elemen-elemen yang membentuk sebuah sistem :

1. Tujuan

   Setiap sistem memiliki tujuan (*Goal*), entah hanya satu atau mungkin banyak. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Tanpa tujuan, sistem menjadi tak terarah dan tak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem yang lain berbeda.

2. Masukan

   Masukan *(input)* sistem adalah segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan yang diproses. Masukan dapat berupa hal-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 merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna dan lbih bernilai, misalnya berupa informasi dan produk, tetapi juga bisa berupa hal-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)* merupakan hasil dari pemrosesan. Pada sistem informasi, keluaran bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

5. Batas

   Yang disebut batas *(boundary)* sistem adalah pemisah antara sistem dan daerah di luar sistem (lingkungan). Batas sistem menentukan konfigurasi, ruang lingkup, atau kemampuan sistem. Sebagai contoh, tim sepakbola mempunyai aturan permainan dan keterbatasan kemampuan pemain. Pertumbuhan sebuah toko kelontong dipengaruhi oleh pembelian pelanggan, gerakan pesaing dan keterbatasan dana dari *bank*. Tentu saja batas sebuah sistem dapat dikurangi atau dimodifikasi sehingga akan mengubah perilaku sistem. Sebagai contoh, dengan menjual saham ke publik, sebuah perusahaan dapat mengurangi keterbasatan dana.

6. Mekanisme Pengendalian dan Umpan Balik

   Mekanisme pengendalian (*control mechanism*) diwujudkan dengan menggunakan umpan balik (*feedback*), yang mencuplik keluaran. Umpan balik ini digunakan untuk mengendalikan baik masukan maupun proses. Tujuannya adalah untuk mengatur agar sistem berjalan sesuai dengan tujuan.

7. Lingkungan

Lingkungan adalah segala sesuatu yang berada diluar sistem. Lingkungan bisa berpengaruh terhadap operasi sistem dalam arti bisa merugikan atau menguntungkan sistem itu sendiri. Lingkungan yang merugikan tentu saja harus ditahan dan dikendalikan supaya tidak mengganggu kelangsungan operasisistem, sedangkan yang menguntungkan tetap harus terus dijaga, karena akan memacu terhadap kelangsungan hidup sistem.

2.5 Sistem *Online*

Secara umum, sesuatu dikatakan *online* adalah bila terkoneksi/terhubung dalam suatu jaringan ataupun sistem yang lebih besar. Beberapa arti kata *online* lainnya yang lebih spesifik yaitu :

1. Dalam percakapan umum, jaringan/*network* yang lebih besar dalam konteks ini biasanya lebih mengarah pada internet, sehingga '*online*' lebih pada menjelaskan status bahwa ia dapat diakses melalui internet.
2. Secara lebih spesifik dalam sebuah sistem yang terkait pada ukuran dalam satu aktivitas tertentu, sebuah elemen dari sistem tersebut dikatakan *online* jika elemen tersebut beroperasional. Sebagai contoh, sebuah instalasi pembangkit listrik dikatakan *online* jika ia dapat menyediakan listrik pada jaringan elektrik.
3. Dalam telekomunikasi, istilah *online* memiliki arti lain yang lebih spesifik. Suatu alat diasosiasikan dalam sebuah sistem yang lebih besar dikatakan *online* bila berada dalam kontrol langsung dari sistem tersebut. Dalam arti jika ia tersedia saat akan digunakan oleh sistem (*on-demand*), tanpa membutuhkan intervensi manusia, namun tidak bisa beroperasi secara mandiri di luar dari sistem tersebut.

Dengan Internet kita dapat menerima dan mengakses informasi dalam berbagai format dari seluruh penjuru dunia. Kehadiran system *online* juga dapat memberikan kemudahan dalam dunia pendidikan, hal ini terlihat dengan begitu banyaknya situs web yang menyediakan media pembelajaran yang semakin interaktif serta mudah untuk dipelajari.

2.6 Repository

2.6.1. Pengertian *Repository*

Secara sederhana arti dari *repository* adalah tempat pengarsipan. Dalam konteks kepustakawanan *repository* adalah suatu tempat dimana dokumen, informasi atau data disimpan, dipelihara dan didigunakan. Kadang-kadang istilah *depository* dipakai untuk menyatakan hal yang sama. (Reitz, 2004) menyatakan bahwa *repository* adalah ruang fisik (bangunan, ruangan, area) disediakan untuk penyimpanan permanen atau menengah bahan arsip (manuskrip, buku langka, dokumen pemerintah, kertas, foto dll).Perpustakaan sebenarnya adalah sebuah *repository* akan tetapi dalam ruang lingkup yang lebih luas.

Dari definisi (Reitz, 2004) di atas, terlihat bahwa dokumen yang dikelola dalam *repository* lebih khusus dari pada yang dikelola di perpustakaan. Penyelenggara *repository* lebih mengkhususkan diri untuk mengelola dokumen yang belum diterbitkan oleh perusahaan penerbitan atau penerbitan komersial. Dokumen yang dikelola oleh penyelenggara *repository* sering juga dinamai dengan sebutan literatur kelabu *(gray literature)* yang dapat berupa dokumen yang khas, buku-buku yang jarang didapatkan di pasar buku, dan juga dokumen yang dihasilkan oleh instansi atau lembaga pemerintah dan sebagainya, sehingga ada yang menyebutnya *local contents*.

Berdasarkan pendapat ini, bahwa tempat penyimpanan bukan lagi dalam bentuk bangunan atau ruangan melainkan dalam sebuah *server* komputer, karena bahan yang disimpan, diorganisasikan dan dilayankan adalah bahan-bahan *digital*. *Repository* dalam hal ini adalah bagian dari perpustakaan *digital*. *Repository* menurut pengertian ini yang umumnya dijumpai pada perguruan tinggi termasuk di Indonesia. Sebuah *repository* dapat berupa:

1. Tempat dimana data disimpan
2. Tempat dimana secara khusus data dalam format *digital* disimpan
3. Tempat dimana *e-print* diletakkan
4. Tempat dimana beberapa *database* atau *file* diletakaan untuk didistibusikan secara jaringan *computer*
5. Tempat dimana sesuatu disimpan yang kemunginan untuk digunakan lagi.

2.6.2. Web Repository

Web repository adalah sebuah *web* yang berfungsi sebagai pengumpulan, penyimpanan, serta pemeliharaan kumpulan alat bantu proses untuk memudahkan mahasiswa untuk mengakses sebuah *web* yang berisikan skripsi.

2.6.3. Teori Pengembangan *Web*

"*Web* adalah merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung ke internet" (Sidik, 2012). Fase-fase model *waterfall* menurut *sommerfille*.

1. Requirements Analysis and Definition

Mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan dibangun. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware, database*, dsb. Tahap ini sering disebut dengan *Project Definition*

2. System and Software Design

Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software.* untuk mengetahui sifat dari program yang akan dibuat, maka para *softtware engineer* harus mengerti tentang domain informasi dari *software*, misalnya fungsi yang dibutuhkan, *user interface*, dsb. dari dua aktivitas tersebut harus didokumentasikan dan ditunjuk kepada *user*. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini juga harus didokemntasikan sebagai konfigurasi dari *software.*

3. Implementation and Unit Testing

Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji dengan baik secara unit.

4. Integration and System Testing

Untuk dapat dimengerti oleh mesin, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding.* Tahap ini merupakan implementasi dari

tahap desain yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*System Setting*).

5. Operation and Maintenance

Sesuatu yang dibuat harus diuji coba. Demikian juga dengan *software.* Semua fungsi-fungsi *software* harus diuji coba agar *software* bebas dari *error*, dan hasilnya harus benar benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

2.6.4. Desain (Perancangan Sistem)

Unified Modelling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (Fowler, 2005). " Bangunan dasar metodologi *Unified Modelling Language (UML)* menggunakan tiga bangunan dasar untuk mendeskripsian sistem/perangkat lunak yang akan dikembangkan (Nugroho, 2005).

Ada 4 *things* dalam *Unified Modelling Language (UML)*, yaitu :

1. *Structural Things*

Merupakan bagian yang relatif statis dalam model *Unified Modeling Language (UML)*. Bagian yang relatif statis dapat berupa elemen-elemen yang bersifat fisik maupun konseptual. Contoh : *class, interface, use cases*.

2. *Behavioral Things*

Merupakan bagian yang dinamis pada model *Unified Modelling Language (UML)*, biasanya merupakan kata kerja dari model *Unified Modelling Language (UML)*, yang mencerminkan perilaku sepanjang ruang dan waktu.

Contah : *state*.

3. *Grouphing Things*

Merupakan bagian pengorganisasian dalam *Unified Modelling Language (UML)*. Dalam penggambaran model yang rumit kadang diperlakukan penggambaran paket yang menyerderhanaan model. Paket-paket ini kemudian

dapat di dekomposisi lebih lanjut. Paket berguna bagi pengelompokkan sesuatu, misalnya model-model dan subsistem. Contoh : *package*.

4. *Annotional things*

Merupakan bagian yang memperjelas model *Unified Modelling Language (UML)* dan dapat berupa komentar-komentar yang menjelaskan fungsi serta ciri-ciri setiap elemen dalam model Unified Modelling Language (UML).

Contoh : *notes*.

- Relasi (Relationship)

Ada 4 macam *relationship* dalam *Unified Modelling Language (UML)*, yaitu:

1. Ketergantungan

   Merupakan hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri *(independent)*.

2. Asosiasi

   Merupakan apa yang menghubungkan antara objek satu dengan objek lainnya, bagaimana hubungan suatu objek dengan objek lainnya. Suatu bentuk asosiasi adalah agregasi yang menampilkan hubungan suatu objek dengan bagian-bagiannya.

3. Generalisasi

   Merupakan hubungan dimana objek anak berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk. Arah dari atas kebawah dari induk ke objek anak dinamakan spesialisasi, sedangkan arah berlawanan sebaliknya dari arah bawah keatas dinamakan generalisasi.

4. Realisasi

   Merupakan operasi yang benar-benar dilakukan oleh suatu objek.

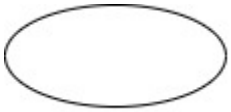
2.6.5. Diagram

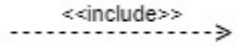
Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsian interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan.

Skenario adalah rangkaian langkah-langkah yang menjabarkan sebuah interaksi antara *user* dengan sebuah sistem, dalam bahasa *use case user* disebut sebagai aktor setiap langkah-langkah dalam *use case* adalah sebuah elemen dalam interaksi antara aktor dan sistem.

Masalah yang biasa terjadi dengan *use case* adalah dengan memusatkan perhatian pada interaksi antara *user* dan sistem, perlu diingat bahwa *use case* mewakili sebuah gambaran eksternal sebuah sistem oleh karena itu jangan mengharapkan adanya korelasi antara *use case* dan *class* di dalam sistem. Berikut adalah simbol-simbol *use case* diagram :

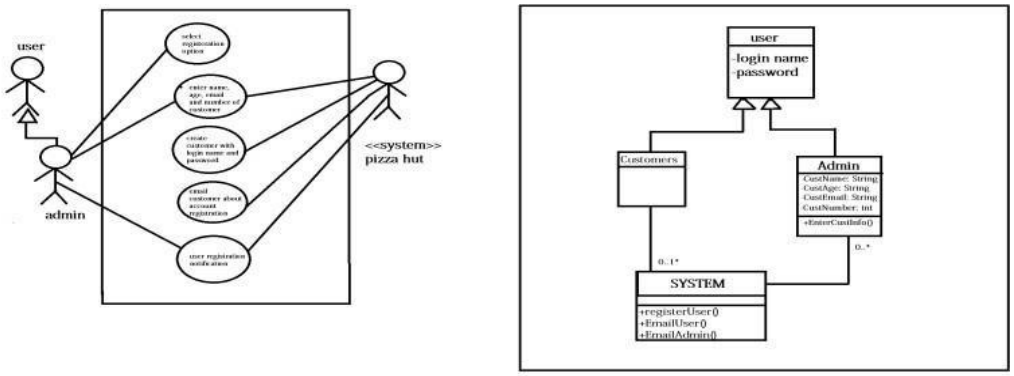
Tabel 2.1 Simbol-Simbol *Use Case* Diagram

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan system sebagai unit-unit yang bertukar pesan antar *unit* dengan aktif, yang dinyatakan dengan menggunakan kata kerja.
	Actor atau actor adalah abstraction dari orang atau system yang lain yang mengaktifkan fungsi dari target system untuk mengidentifikasikan actor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target system orang atau system bisa muncul dalam beberapa peran. Perlu dicatat bahwa actor berinteraksi dengan *use case* tetapi tidak memiliki *control* terhadap *use case*.
	Asosiasi anatra *actor* dan *use case*, digambarkan dengan garis tanpa penuh yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.

	Asosiasi antara *actor* dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan *system*.
	*Include*, merupakan didalam *use case* lain(*required*) atau pemanggilan *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

Sumber : (Hendini, 2016)

Dibawah ini adalah contoh gambar *use case* diagram setelah di aplikasi dari simbol-simbol diatas :



Gambar 2.2 *Use Case* Diagram

2.6.6. *Sequence* Diagram

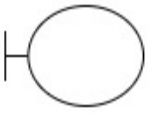
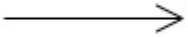
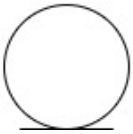
Sequence Diagram adalah menjabarkan *behavior* sebuah skenario tunggal seperti menunjukkan jumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case*. Di dalam *sequence* diagram menunjukkan beberapa notasi tambahan untuk membuat dan menghapus partisipan. Untuk membuat partisipan hanya membuat sebuah tanda panah pesan langsung ke kotak partisipan sedangkan untuk penghapusan sebuah partisipan dengan menandai menggunakan tanda X besar garis pesan yang melewati tanda tersebut menunjukkan satu partisipan menghapus dirinya sendiri.

Komponen *Sequence* Diagram

- *Actor*
- *Interface (Boundary)*
- Proses pembacaan (*Control*)
- Nama *table (Entity)*

Berikut simbol-simbol dari *Sequence* Diagram :

Tabel 2.2 Simbol *Sequence* Diagram

No.	Gambar	Nama	Keterangan
1.		*Actor*	Menggambar orang yang sedang berinteraksi dengan *system*.
2.		*Control Class*	Menggambarkan penghubung antara *boundary* dengan *table*.
3.		*Boundary Class*	Menggambarkan sebuah gambaran dari *form*.
4.		*A massage*	Menggambarkan pengiriman pesan.
5.		*A Focus of Control & A Life Line*	Menggambarkan tempat mulai dan berakhirnya *massage*.
6.		*Entity Class*	Menggambarkan hubungan yang akan dilakukan.

- Partisipan : objek atau entitas yang bertindak dalam *sequence* diagram
- *Message* : Komunikasi antar objek partisipan
- Terdapat 2 tipe garis yaitu vertikal dan horisontal

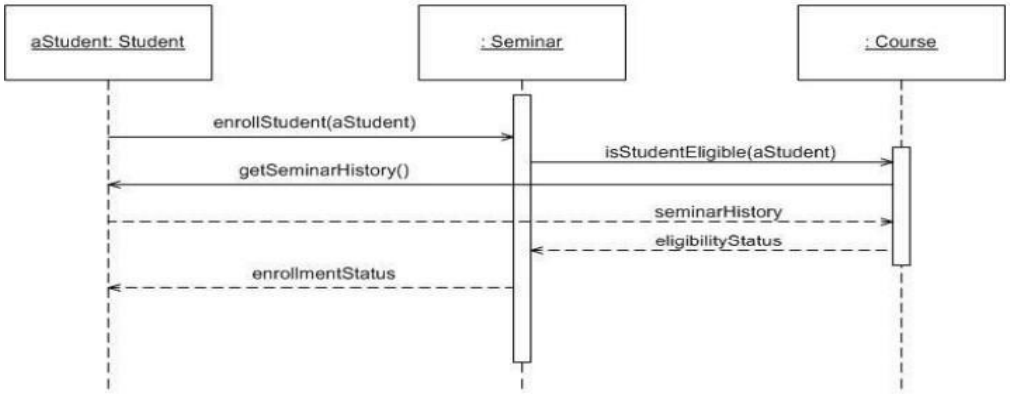
1. Vertikal : waktu ⟶ maju berdasarkan waktu

2. Horisontal : objek mana yang beraksi

- Nama Objek / Class

1. Nama bersifat *optional*.
2. *Boxes* berupa objek diberikan tanda garis bawah.
3. Objek yang tidak bernama disebut *anonymous* objek.
4. *Boxes* berupa *actor* dapat juga digambarkan dengan *stick figure*.

Berikut contoh gambar *sequence* diagram seperti pada Gambar 2.3.



Gambar 2.3 *Sequence* Diagram

3. *Class* Diagram

Class Diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antara mereka. *Class* diagram juga menunjukkan properti dan operasi sebuah *class* (Fowler, 2005). Notasi *class* dapat dilihat pada table.

Tabel 2.3 Notasi Class

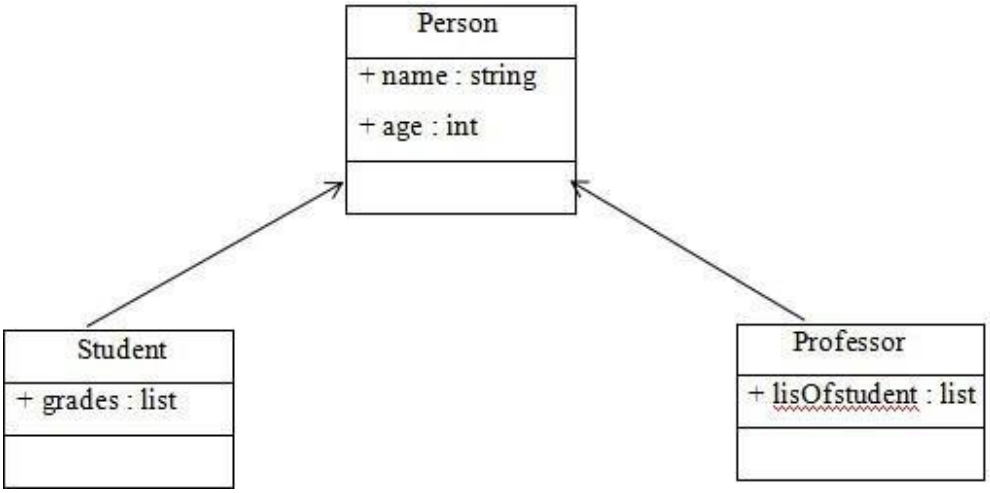
Nama Kelas
Atribut
Operasi

(Sumber: Nugroho, 2005)

Bagian paling atas memuat nama kelas. Bagian tengah mendaftarkan atribut-atribut yang dimiliki sebuah kelas sedangkan paling bawah mendaftarkan operasi operasi yang dimiliki kelas yang bersangkutan (Nugroho, 2005).

Class diagram umumnya tersusun dari elemen *class, interface, dependency*. *Generalization* dan *Association*. Relasi *dependency* menunjukkan bagaimana terjadi

ketergantungan antar *class* yang ada. Relasi *Generalization* menunjukkan bagaimana suatu *class* menjadi *superclass* dari *class* lainnya dan *class* tersebut menjadi *subclass* dari *class* tersebut. Relasi *Association* menggambarkan navigasi antar *class*, berapa banyak objek lain bisa berhubungan dengan suatu objek, dan apakah satu *class* menjadi bagian dari *class* lainnya (Hermawan, 2004). Contoh *Class* Diagram dapat dilihat pada Gambar 2.4.



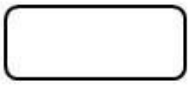
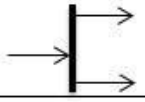
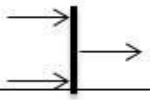
Gambar 2.4 Class Diagram

4. *Activity* Diagram

Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram hampir sama dengan sebuah aliran data, akan tetapi perbedaan prinsip antara *activity* diagram dengan notasi diagram alir adalah pada *activity* diagram lebih mendukung kepada *behavior parallel*.

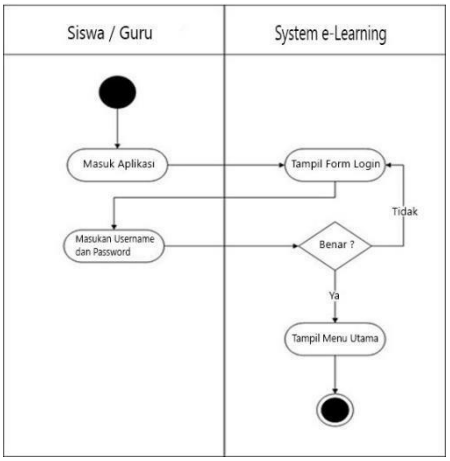
Activity diagram telah mengalami beberapa perubahan paling besar selama perkembangan versi-versi *UML*, jadi tidaklah mengejutkan jika *activity* diagram telah dikembangkan secara *significant* dan diubah lagi kedalam bentuk *UML* 2. Dianggap sebagai kasus khusus *state* diagram, karena menyebabkan banyaknya masalah bagi pengguna yang memodelkan jalur kerja yang mana lebih cocok dikerjakan oleh *activity* diagram.

Simbol-simbol *activity* diagram dapat dilihat pada gambar sebagai berikut :

Start point (Poin Awal)	
End point (Point Akhir)	
Activitiest (Aktivitas Proses)	
Fork (Pecabanga)	
Join (Penggabungan)	
Decession	
Swimlane	Sebuah cara untuk mengelompokan aktivitas

Gambar 2.5 Simbol-simbol *Activity* diagram

Contoh *activity* diagram dapat dilihat pada Gambar 2.6 :



Gambar 2.6 Activity Diagram

2.6.5. *PHP*

Menurut (Sidik 2012) *PHP* (*HyperText Preprocessor*) adalah sebuah bahasa utama *script server side* yang disisipkan pada *HTML* yang dijalankan di *server*, dan juga bisa digunakan untuk membuat aplikasi *desktop*.

Contoh *Script PHP* :

<?*php*

//semua kode *PHP* diletakkan disini ;

?

Dalam variabel *PHP* terdapat beberapa variabel yang digunakan pada penulisan kode program, adalah sebagai berikut :

1. Dalam *PHP* variabel dimulai oleh simbol dolar ($) diikuti oleh nama variabel, seperti berikut :

   $nama_variabel = Nilai;

2. Terdapat beberapa aturan yang harus diketahui saat menentukan nama variabel *PHP*, yaitu :

a. Variabel harus dimulai huruf dan boleh diikuti oleh huruf atau bilangan dan garis bawah.
b. Variabel hanya dapat terdiri dari alfanumerik (a..z,A..Z,0..9) dan garis bawah (_).
c. Variabel dengan lebih dari satu kata sebaiknya dipisahkan dengan garis bawah, misalnya $status_perkawinan, bukan spasi.

Sedangkan menurut (Nugroho 2006) "*PHP* atau singkatan dari *Personal Home Page* merupakan bahasa *script* yang tertanam dalam *HTML* untuk dieksekusi bersifat *server side*". *PHP* juga dapat berjalan pada berbagai *web server* seperti *IIS* (*Internet Information Server*), *PWS* (*Personal Web Server*), *Apache*. *PHP* juga mampu berjalan di banyak sistem operasi yang beredar saat ini, diantaranya: Sistem Operasi *Microsoft Windows* (semua versi), *Linux, Mac Os. PHP* dapat dibangun sebagai modul *web server Apache* dan sebagai *binary* yang dapat berjalan sebagai *CGI* (*Common Gateway Interface*).

PHP dapat mengirim *HTTP header*, dapat mengatur *cookies*, mengatur *authentication* dan *redirect user*. Salah satu keunggulan yang dimiliki *PHP* adalah kemampuannya untuk melakukan koneksi ke berbagai macam *software* sistem manajemen *basis data* atau *Database Management Sistem* (*DBMS*), sehingga dapat menciptakan suatu halam *web* dinamis.

PHP mempunyai koneksitas yang baik dengan beberapa *DBMS* seperti *Oracle*, *MySQL, Microsoft SQL Server, Solid, PostgreSQL, File Pro,* dan tidak terkecuali semua *database* ber-interface *ODBC*.

Hampir seluruh aplikasi berbasis *web* dapat dibuat dengan *PHP*. Namun kekuatan utama adalah konektivitas *basis data* dengan *web*. Dengan kemampuan ini kita akan mempunyai suatu sistem *basis data* yang dapat diaksees.

2.6.6. *MySQL*

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengolahan datanya (Rudianto, 2011). *MySQL* merupakan *software database open source*, *MySQL* menjadi pilihan utama bagi banyak pengembang *software* dikarenakan kelebihan *MySQL* didukung programprogram umum seperti *C, C++, Java, PHP,* dan *Python*.

Proses menggunakan *MySQL* pada dasarnya adalah mengelola data dan informasi agar data dan informasi tersimpan dengan tertata dan rapi, proses-proses yang sering terjadi biasanya adalah membuat *database*, membuat sebuah *table*, memodifikasi struktur sebuah *table*, mengisikan data dalam sebuah *table*,menghapus data dalam sebuah *table*, memodifikasi (merubah atau mengedit) data dalam sebuah *table* dan mencari data dalam sebuah *table* (Saputra, 2012).

1. Tipe Data *MySQL*

MySQL memiliki banyak tipe data berbeda yang dibagi menjadi 3 kategori yaitu *Numeric, Date and Time* dan tipe data *string*. Menentukan jenis dari tipe data merupakan suatu hal yang sangat penting dalam pembuatan tabel, supaya ruang *memory* yang digunakan sesuai dengan kebutuhan.

a. Tipe Data Numerik

Numerik adalah salah satu bentuk data yang berupa data angka.

Dapat dilihat pada Tabel 2.4 Tipe Data Numerik.

Tabel 2.4 Tipe Data Numerik

Tipe Data	Bytes	Keterangan
``	1	Tipe ini merupakan bentuk *numeric* yang paling kecil dalam menangani data di dalamnya, yaitu data dari angka -128 sampai dengan 127.
SMALL INT ()	2	Tipe *small int* dapat menyimpan data lebih besar yaitu mulai dari -32768 sampai dengan 32767.
MEDIUMINT()	3	*MEDIUMINT* mampu menangani data mulai dari -8388608 sampai dengan 8388607.
INT()	4	Tipe ini sering digunakan dalam pembuatan database. *INT* mampu menyimpan data mulai dari -214748348 sampai 214748347.
BEGINT()	8	Bentuk terbesar dalam tipe data numerik. *BEGINT* mampu menangani data mulai dari -9223372036854775808 sampai dengan 9223372036854775807.

b. Tipe Data *Date and Time*

MySQL memiliki beberapa tipe data yang tersedia untuk menampilkan tanggal dan waktu. Dapat dilihat pada Tabel 2.5.

Tabel 2.5 Tipe Data *Date and Time*

Tipe Data	Keterangan
DATE TIME	Tipe ini dapat menyimpan dua buah bentuk data sekaligus, yaitu penanggalan dan waktu. Bentuknya adalah ‘0000-00-00
	00:00:00’.
DATE	Tipe data ini digunakan untuk menyimpan data penanggulan saja. Bentuknya adalah ‘0000-00-00’.
TIMESTAMP	Tipe data *TIMESTAMP* tidak ada pembatasnya, berikut contoh penulisannya 00000000000000.

TIME	Tipe ini digunakan untuk menyimpan data berbentuk penanggalan yaitu mulai dari tahun yang dibaca dari dua karakter terakhir dan diikuti bulan dan tanggal. Bentuknya ’00:00:00’. Contoh 08:35:55
YEAR	Tipe ini hanya menyimpan data berupa tahunnya saja. Bentuknya adalah 0000, contoh 2009.

c. Tipe Data *String*

Walaupun tipe *numeric* dan *date* sangat penting, namun kebanyak dari tipe data yang akan digunakan berada di format *string*.

Tabel 2.6 Tipe Data *String*

Tipe Data	Keterangan
CHAR ()	Tipe ini sama dengan *VARCHAR*, yaitu dapat menyimpan data sampai dengan 225 karakter.
VARCHAR	Tipe ini dapat menyimpan data sampai dengan 225 karakter.
TINYTEXT	Tipe ini merupakan bentuk terkecil dari penyimpanan data string, tipe ini mampu menangani data sampai dengan 2^8-1 data.
BLOB	Tipe ini mampu menangani data sampai dengan 2^16-1 (64K-1) data.
TINYBLOB	Tipe ini sama dengan *TINYTEXT*, yaitu menangani data sampai dengan 2^8-1 data.
TEXT	Bentuk *TEXT* adalah salah satu bentuk dukungan tipe *string* yang mampu menangani data sampai dengan 2^16-1 (64K-1)
	data.
MEDIUMTEXT	Tipe ini dapat menyimpan data yang cukup besar, yaitu sampai dengan 2^24-1 (16M-1) data.
ENUM	Tipe ini merupakan tipe yang dikatakan sebagai tipe validasi, pada tipe ini yang mungkin akan menjadi isi dari kolom tersebut harus ditentukan terlebih dahulu.

SET	Tipe ini memiliki fungsi yang sama dengan tipe *ENUM*, yaitu dengan mendeklarasikan anggota dari ini kolom yang mungkin akan menjadi anggotanya.

2.6.7. Fungsi *PHP MySQL*

1. *Mysql_connect*()

Fungsi pertama kali untuk dapat terhubung ke *MySQL* ialah fungsi mysql_connect(). Fungsi ini mempunyai atribut lengkap, yaitu :

Mysql_connect("$*host","*$*username","*$*password");*

Tabel 2.7 Atribut *Mysql_connect*()

$*host*	*Hostname/IP Addres* yang digunakan untuk mengakses *MySQL*
$*username*	*User* yang mempunyai account *MySQL*
$*password*	*Password*

2. *Mysql_select_db*()

Sesudah terhubung ke mysql, langkah selanjutnya ialah memilih *database* yang akan digunakan.

Mysql_select_db($*db*,$*link_id*)

Tabel 2.8 Atribut *Mysql_select_db*()

$*db*	Nama *database*, contoh *php*
$*link_id*	Variabel untuk terhubung ke *mysql*, dalam hal ini (lihat *connect.php*) ialah $koneksi,

3. *Mysql_query*()

Syarat utama untuk mengakses mysql sudah terpenuhi. Fungsi ketiga ialah fungsi untuk melakukan *query* ke *mysql*.

Mysql_query($query,$*link_id*)

4. *Mysql_num_rows*()

Fungsi ini digunakan untuk menghitung banyak baris yang diambil dari variabel *query*.

Mysql_num_rows($*query*)

5. *Mysql_feteh_array*()

Fungsi dari *php* yang terakhir digunakan untuk mengambil (*fetch*) *record* dari suatu *query*. Fungsi ini menghasilkan nilay *array*. Dengan fungsi ini, hasil *query* dapat ditampilkan di *browser*. *Mysql_fetch_array*($*query*)

2.6.8. *XAMPP*

XAMPP adalah sebuah aplikasi *web server* instan dan lengkap dikarenakan segala yang anda butuhkan untuk membuat sebuah situs *web* dengan *Content Management System* (Joomla) bisa dicoba di dalam aplikasi ini. *XAMPP* adalah sebuah paket *installer AMP* (*Apache, MySQL,* dan *PHP*) yang sangat mudah untuk diaplikasikan dala komputer anda yang belum memiliki *server* untuk dapat melihat situs yang anda buat menggunakan Bahasa *server* dan *database server* tersebut.

Menurut Wahana(2009) "*XAMPP* adalah salah satu paket instalasi *apache, PHP*, dan *MySQL* secara instant yang dapat digunakan untuk membantu proses instalasi ketiga produk tersebut"

2.6.9. *Code igniter*

Menurut Betha Sidik (2012) *CodeIgniter* adalah :" Sebuah *framework php* yang bersifat *open source* dan menggunakan metode *MVC* (*Model, View, Controller*) untuk memudahkan *developer* atau *programmer* dalam membangun sebuah aplikasi berbasis *web* tanpa harus membuatnya dari awal".

Dalam situs resmi *code igniter*, (*Official Website CodeIgniter*,2002) menyebutkan bahwa *code igniter* merupakan *framework PHP* yang kuat dan sedikit *bug*. *Code igniter* ini dibangun untuk para pengembang dengan bahasa pemrograman *PHP* yang membutuhkan alat untuk membuat *web* dengan fitur lengkap.

Framework Code igniter dikembangkan oleh Rick Ellis, *CEO* Ellislab, *Inc.* Kelebihan dari *framework code igniter* jika dibandingkan dengan *framework* lain adalah sebagai berikut :

1. *Gratis (Open-Source*)

   Kerangka kerja *Code igniter* memiliki lisensi dibawah *Apache/BSD open-source* sehingga bersifat bebas atau *gratis.*

2. Berukuran kecil

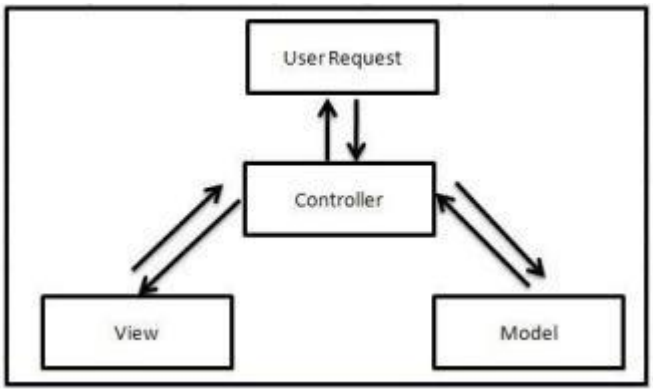
   Ukuran yang kecil merupakan keunggulan tersendiri jika dibandingkan *framework* lain yang berukuran besar dan membutuhkan *resource* yang besar dan juga dalam eksekusi maupun penyimpanannya.

3. Menggunakan konsep *M-V-C Code igniter* merupakan konsep *M-V-C* (*Model-ViewController*) yang memungkinkan pemisahan antara *layer application-logic* dan *presentation*. Dengan konsep ini kode *PHP*, *query Mysql*, *Java script* dan *CSS* dapat saling dipisah-pisahkan sehingga ukuran *file* menjadi lebih kecil dan lebih mudah dalam perbaikan kedepannya atau *maintenance*.

   a. *Model*. Kode merupakan *program* (berupa *OOP class*) yang digunakan untuk berhubungan dengan *database MySQL* sekaligus untuk memanipulasinya (*input edit-delete*).

   b. *View*. Merupakan kode *program* berupa *template* atau *PHP* untuk menampilkan data pada *browser*.

   c. *Controller*. Merupakan Kode *program* (berupa *OOP class*) yang digunakan untuk mengontrol aliran atau dengan kata lain sebagai pengontrol *model* dan *view*. Adapun alur dari program aplikasi berbasis *code igniter* yang menggunakan konsep *M-V-C* ditunjukkan pada gambar berikut :

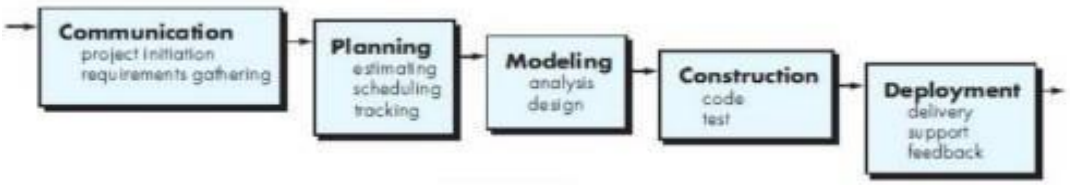


Gambar 2.7 Konsep M-V-C

2.6.10. Metode *Waterfall*

Metode *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya adalah "*Linear Sequential Model*". Model ini sering disebut juga dengan *"classic life cycle"* atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali

diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, Tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman, 2015). Fase-fase dalam *Waterfall Model* menurut referensi (Pressman, 2015) :



Gambar 2.8 Metode *Waterfall*

Dengan demikian, metode *waterfall* dianggap pendekatan yang lebih cocok digunakan untuk *project* pembuatan sistem baru dan juga pengembangan *software* dengan tingkat resiko yang kecil serta waktu pengembangan yang cukup lama. Tetapi salah satu kelemahan paling mendasar adalah menyamakan pengembangan *hardware* dan *software* dengan meniadakan perubahan saat pengembangan. Padahal, *error* diketahui saat *software* dijalankan, dan perubahan-perubahan akan sering terjadi.

Keuntungan menggunakan metode *waterfall* adalah prosesnya lebih terstruktur, hal ini membuat kualitas *software* baik dan tetap terjaga. Dari sisi *user* juga lebih menguntungkan, karena dapat merencanakan dan menyiapkan kebutuhan data dan proses yang diperlukan sejak awal. Penjadwalan juga menjadi lebih menentu, karena jadwal setiap proses dapat ditentukan secara pasti. Sehingga dapat dilihat jelas target penyelesaian pengembangan program. Dengan adanya urutan yang pasti, dapat dilihat pula perkembangan untuk setiap tahap secara pasti. Dari sisi lain, model ini merupakan jenis model yang bersifat dokumen lengkap sehingga proses pemeliharaan dapat dilakukan dengan mudah.

Kelemahan menggunakan metode *waterfall* adalah bersifat kaku, sehingga sulit melakukan perubahan di tengah proses. Jika terdapat kekurangan proses/prosedur dari tahap sebelumnya, maka tahapan pengembangan harus dilakukan mulai dari awal lagi. Hal ini akan memakan waktu yang lebih lama. Karena jika proses sebelumnya belum selesai sampai akhir, maka proses selanjutnya juga tidak dapat berjalan. Oleh karena itu, jika terdapat kekurangan dalam permintaan *user* maka proses pengembangan harus

dimulai kembali dari awal. Karena itu, dapat dikatakan proses pengembangan *software* dengan metode *waterfall* bersifat lambat.

Kelemahan lainnya menggunakan metode *waterfall* adalah membutuhkan daftar kebutuhan yang lengkap sejak awal. Untuk menghindari pengulangan tahap dari awal, *user* harus memberikan seluruh prosedur, data, dan laporan yang diinginkan mulai dari tahap awal pengembangan. Tetapi pada banyak kondisi, *user* sering melakukan permintaan di tahap pertengahan pengembangan sistem. Dengan metode ini, maka *development* harus dilakukan mulai lagi dari tahap awal. Karena *development* disesuaikan dengan desain hasil *user* pada saat tahap pengembangan awal. Di sisi lain, *user* tidak dapat mencoba sistem sebelum sistem benar-benar selesai. Selain itu, kinerja *personal* menjadi kurang *optimal* karena terdapat proses menunggu suatu tahap selesai terlebih dahulu. Oleh karena itu, seringkali diperlukan *personal* yang "*multi-skilled*" sehingga minimal dapat membantu pengerjaan untuk tahapan berikutnya. (Pressman, 2015).

2.7 Pengujian *Black Box*

Menurut Iskandaria (2012), Pengujian *black box* (*black box testing*) adalah salah satu metode pengujian perangkat lunak yang berfokus pada sisi fungsionalitas, khususnya pada *input* dan *output* aplikasi (apakah sudah sesuai dengan apa yang diharapkan atau belum). Tahap pengujian merupakan salah satu tahap yang harus ada dalam sebuah siklus pengembangan perangkat lunak.

Menurut Shihab (2011), *Black Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, *tester* dapat mendefinisikan kumpulan kondisi *input* dan melakukan pengetesan pada spesifikasi fungsional *program*.

Shihab (2011), mengemukakan ciri-ciri *black box testing*, yaitu:

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

Dengan adanya pengujian *black box testing* ini diharapkan jika ada kesalahan maupun kekurangan di dalam aplikasi dapat segera diketahui sekecil mungkin oleh peneliti.

2.8 *Entity Relationship Diagram (ERD)*

Menurut Utami dan Hartanto (2012), *Entity Relationship Diagram* (*ERD*) adalah suatu diagram untuk menggambarkan desain konseptual dari model suatu *basis* data relasional. *ERD* juga merupakan gambaran yang menghubungkan antara objek satu dengan objek 12 yang lain dalam dunia nyata. Bisa dikatakan bahwa bahan yang akan digunakan untuk membuat *ERD* adalah dari objek dunia nyata.

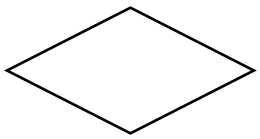
a. Komponen *Entity Relationship Diagram* (*ERD*)

Adapun bentuk *symbol* grafis atau komponen dari *Entity Relationship Diagram* (*ERD*) sebagai berikut (Priyadi, 2014):

1. Entitas. Entitas merupakan notasi untuk mewakili suatu objek dengan karateristik sama, yang dilengkapi oleh atribut, berfungsi memberikan identitas sehingga pada suatu lingkungan nyata setiap objek akan berbeda dengan lainya. Entitas di lambangkan dengan bentuk persegi panjang sepeti Gambar 2.9.

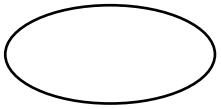
Gambar 2.9 Notasi Entitas

2. Relasi. Relasi merupakan notasi yang digunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan. Dan fungsinya untuk mengetahui jenis hubungan antara 2 *file*, relasi berbentuk belah ketupat seperti Gambar 2.10.



Gambar 2.10 Notasi Relasi

3. Atribut. Berbentuk lingkaran atau elip yang menyatakan atribut (atribut yang berfungsi sebagai *key* digaris bawahi). Atribut digambarkan pada Gambar 2.11.



Gambar 2.11 Notasi Atribut

4. Garis Penghubung. Garis penghubung merupakan notasi yang menghubungkan antara atribut dengan entitas dan entitas dengan relasi yang digunakan dalam Diagram E-R. Garis penghubung berbentuk garis seperti tertera pada gambar 2.12.

Gambar 2.12 Notasi Garis Penghubung

b. Kardinalitas atau Derajat Relasi

Menurut Fathansyah (2012), Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada entitas yang lain. Terdapat beberapa jenis kardinalitas yang menggambarkan relasi antar entitas, adapun beberapa kardinalitas tersebut sebagai berikut:

1. Relasi satu-ke-satu *(one-to-one)*

   Entitas hanya boleh berhubungan dengan satu entitas kedua dan sebaliknya yang dijelaskan pada Gambar 2.13.

Gambar 2.13 Relasi *one-to-one*

2. Relasi satu-ke-banyak *(one-to-many)*

   Entitas pertama boleh banyak berhubungan dengan entitas kedua, tetapi entitas kedua hanya boleh berhubungan dengan satu entitas atau sebaliknya yang dijelaskan pada Gambar 2.14.

Gambar 2.14 Relasi *one-to-many*

3. Relasi banyak-ke-satu *(many-to-one)*

   Entitas pertama hanya berhubungan dengan satu entitas kedua, tetapi entitas kedua boleh berhubungan dengan banyak entitas atau sebaliknya yang dijelaskan pada Gambar 2.15.

Gambar 2.15 Relasi *many-to-one*

4. Relasi banyak-ke-banyak *(many-to-many)*

   Entitas pertama boleh banyak berhubungan dengan entitas kedua atau sebaliknya yang dijelaskan pada Gambar 2.16.

Gambar 2.16 Relasi *many-to-many*