BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Sistem diartikan sebagai kumpulan dari komponen yang saling berkaitan untuk secara bersama-sama menghasilkan satu tujuan. Mengenai hirarki pengelompokkannya, dapat dikemukakan bahwa apabila suatu komponen di dalam suatu sistem membentuk sistem sendiri maka komponen ini dinamakan subsistem dan seterusnya sehingga akan ada nama-nama modul, submodul, aplikasi dan subaplikasi. Hirarki ini berlaku relatif, tergantung dari jenjang manajerial manakah dimulainya (Sutarbi, 2003).

Menurut Sutarbi (2003) sistem adalah setiap kumpulan dari komponen atau sub-sistem yang berinteraksi untuk mencapai suatu tujuan tertentu. Informasi diartikan sebagai hasil pengolahan data yang digunakan untuk suatu keperluan, sehingga penerimanya akan mendapat rangsangan untuk melakukan tindakan. Data adalah fakta yang jelas lingkup, tempat dan waktunya. Data diperoleh dari sumber data primer atau sekunder dalam bentuk berita tertulis atau sinyal elektronis. Pengertian informasi dan data berlaku sangat relative tergantung pada posisinya terhadap lingkup permasalahannya. Jenis-jenis informasi dapat dipandang dari tiga segi yaitu manajerial, sumber dan rutinitasnya. Dari segi manajerialnya dibagi tiga jenis:

1. Informasi Strategis.
2. Informasi Taktis.
3. Informasi Operasional.

Informasi strategis adalah informasi yang digunakan untuk kegiatan manajerial tingkat atas (*top* manajemen) dan umumnya mempunyai daya jangkau untuk waktu 5 sampai 15 tahun bahkan mungkin 75 tahun. Informasi taktis digunakan untuk manajerial tingkat menengah (*midle* manajemen) pada umumnya dengan daya jangkau satu tahun. Sedangkan informasi operasional adalah informasi yang digunakan oleh kegiatan manajerial tingkat bawah (*low*

manajerial) dan pada umumnya mempunyai daya jangkau dalam hitungan beberapa hari (Sutarbi, 2003).

Informasi dilihat dari sumbernya dibagi menjadi dua jenis internal dan eksternal. Informasi internal adalah informasi yang menggambarkan keadaan (*profile*), dan informasi eksternal adalah informasi yang menggambarkan ada tidaknya perubahan di luar organisasi itu. Informasi eksternal lebih banyak digunakan oleh kegiatan manajerial tingkat atas. Jenis informasi dibagi menjadi informasi insendentil dan rutin. Informasi rutin digunakan secara periodik terjadwal dan digunakan untuk penanggulangan masalah-masalah rutin, infomasi insendentil diperlukan untuk penanggulangan masalah-masalah khusus (Sutarbi, 2003).

Sistem informasi secara teknis dapat didefinisikan sebagai sekumpulan komponen yang saling berhubungan, mengumpulkan atau mendapatkan, memproses, menyimpan dan mendistribusikan informasi untuk menunjang pengambilan keputusan serta pengawasan dalam suatu organisasi. Selain menunjang proses pengambilan keputusan, koordinasi dan pengawasan, sistem informasi juga dapat membantu manajer dan karyawan menganalisis permasalahan, menggambarkan hal-hal yang rumit dan menciptakan produk baru (Sutarbi, 2003).

Pengertian sistem informasi dapat dilihat dari segi fisik dan fungsinya. Segi fisiknya dapat diartikan susunan yang terdiri dari perangkat keras, perangkat lunak dan tenaga pelaksananya yang secara bersama-sama saling mendukung untuk menghasilkan suatu produk. Sedangkan dari segi fungsi informasi merupakan suatu proses berurutan dimulai dari pengumpulan data dan diakhiri dengan komunikasi atau desiminasi. Selanjutnya sistem informasi dikatakan berdaya guna jika mampu menghasilkan informasi yang baik, tinggi akurasinya, tepat waktu, lengkap dan ringkas isinya. Akurasi adalah ukuran berupa rasio antara jumlah informasi yang benar dan tidak benar. Suatu sistem dikatakan mempunyai akurasi tinggi apabila akurasinya sebesar 95%, namun akurasi tinggi tidak akan berguna apabila kedatangannya terlambat dan tidak teratur. Oleh

karena itu sistem informasi dituntut untuk lengkap, ringkas dan teratur sehingga tidak memusingkan pengguna informasi tersebut (Sutarbi, 2003).

Lingkungan perpustakaan ada tiga yaitu:

1. Pengelola.

   Suatu sistem informasi dapat diselenggarakan apabila ada suatu unit kerja yang diberi tanggug jawab untuk mengelolanya. Tugas pengelola ini adalah melaksanakan koordinasi dalam pengembangan, pemeliharaan dan pengoperasian, melayani permintaan data, pengembangan teknik atau metode analisis dalam rangka pendayagunaan informasi, dan bertanggung jawab atas semua kualitas data dan informasi yang dihasilkan.

2. Kepekaan.

   Sistem informasi dapat berguna apabila memberi layanan sesuai dengan apa yang seharusnya diperlukan. Kemudian diperlukan pembaruan agar penyusunan informasi sesuai dengan keadaan lapangan. Suatu mekanisme yang harmonis antara sumber data dengan pusat penyimpanan data harus saling menguntungkan. Oleh karena itu informasi yang dihasilkan harus mempunyai beragam bentuk dan secara langsung mampu memberikan semacam peringatan kepada penerima informasi tentang adanya faktor-faktor negatif yang perlu segera ditanggulangi.

3. Kesederhanaan.

   Sistem informasi harus tersusun dari serangkaian perangkat keras, perangkat lunak dan juga prosedur yang mudah dimengerti maupun dioperasikan serta dipelihara oleh seluruh unit kerja, agar dapat dihindari kemungkinan kesalahpahaman atau peluang terjadinya penyimpangan. Ada ketentuan yang jelas dan sistematik dalam membantu tersajinya sistem informasi manajemen. Dari semua pengertian dasar dan prinsip-prinsip ini, yang terkandung di dalamnya dapat diartikan bahwa:

   a. *Output* dari sistem informasi adalah informasi. Relevansi dan kualitas informasi yang dihasilkan tergantung sepenuhnya pada keinginan manusia. Sistem informasi harus mengandung empat komponen, yaitu: data, perangkat keras, perangkat lunak dan manusia. Perangkat keras

maupun perangkat lunak hanya merupakan alat bantu yang tidak akan melakukan apapun apabila tidak ada data yang diproses dan tidak ada yang memerintahkan. Ada tiga peranan manusia yang diperlukan oleh sistem informasi yatiu sebagai pemberi data, pengolah dan pengguna data. Ketiga peranan ini merupakan satu kesatuan yang tidak terpisahkan di mana yang satu tidak merasa lebih penting dari yang lain. Peranan ini tidak ada hubungannya dengan jabatan struktural dan berlaku sangat relatif terhadap lingkup permasalahannya.

b. Sistem informasi harus mempunyai kejelasan tujuan dan bukan berarti komputerasasi total. Komputerisasi hanya dikenakan secara selektif terhadap aktivitas-aktivitas yang berhubungan dengan data yang berskala besar tapi memerlukan proses yang menuntut ketelitian dan kecepatan tinggi di mana pekerjan secara manual sudah tidak mungkin dipertahankan.
c. Sistem informasi adalah proses yang berlangsung secara periodik dan beroperasi dalam suatu siklus yang bergerak secara teratur. Oleh karena itu, suatu sistem informasi lebih beroreintasi pada informasi yang bersifat rutin.
d. Sistem informasi memerlukan satu pengelola yang berperanan sebagai Koordinator baik dalam pemeliharaan maupun dalam pengembangannya. Berarti bahwa sistem informasi perlu diwadahi dalam bentuk fungsi tersendiri dari suatu organisasi atau unit kerja. Dari konsepsi teoritis diatas jika dikaitkan dengan pengelolaan perpustakaan maka sistem informasi di perpustakaan harus dikelola oleh tenaga yang professional yang memiliki keahlian dalam menata dan menyimpan *literature* sehingga memudahkan pengunjung dalam mencari *literature* yang diperlukan. Dalam penyimpanan penataan buku sebagaimana perlu diingat aspekaspek kepekaan, dalam arti menata buku harus mampu memberikan pelayanan terbaik bagi para pengunjung.

2.2 Sistem Informasi Perpustakaan

Sistem informasi perpustakaan menurut Gordon (2003) sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data harian, penunjang kegiatan dalam penyimpanan data dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Sistem informasi perpustakaan (SIPERPUS) merupakan perangkat lunak yang didesain khusus untuk mempermudah pendataan koleksi perpustakaan, katalog data anggota atau peminjam, transaksi dan sirkulasi koleksi perpustakaan (SIPERPUS, 2009).

Sistem informasi perpustakaan adalah proses komputerisasi untuk mengolah data suatu perpustakaan. Mulai dari katalogisasi koleksi, pengolahan data anggota, sampai proses peminjaman dan pengembalian koleksi beserta aturan-aturannya seperti lamanya peminjaman dan penghitungan denda keterlambatan. Sistem informasi perpustakaan tidak lengkap tanpa adanya *online public access catalog* (OPAC) atau *intranet public acces Catalog* (IPAC) yaitu suatu katalog yang memuat informasi tentang koleksi yang dimiliki sebuah perpustakaan (Gordon, 2003).

2.2.1 Pengertian Perpustakaan

Perpustakaan adalah institusi atau lembaga yang menyediakan koleksi bahan perpustakaan tertulis, tercetak dan terekam sebagai pusat sumber informasi yang diatur menurut sistem dan aturan yang baku dan digunakan untuk keperluan pendidikan, penelitian dan rekreasi intelektual bagi masyarakat. Perpustakaan secara umum bertujuan untuk melakukan layanan informasi literal kepada masyarakat. Tujuan khusus dibedakan oleh jenis perpustakaannya. Karena tujuannya memberi layanan informasi literal kepada masyarakat maka tugas pokok adalah (Madiwasiu, 2008):

a. Menghimpun bahan pustaka yang meliputi buku dan nonbuku sebagai sumber informasi.
b. Mengolah dan merawat pustaka.
c. Memberikan layanan bahan pustaka.

Perpustakan merupakan bentuk kata dasar dari "pustaka" dalam bahasa sansekerta berarti buku, naskah atau tulisan menurut kamus jawa kuno (kawi) Indonesia. Perpustakkaan dikatakan juga sebagai gedung taman pustaka taman bacaan, pengartian sempit demikian menyebabkan apresiasi terhadap perpustakaan di Indonesia makin rendah (Madiwasiu, 2008).

2.2.2 Jenis-Jenis Perpustakaan

Berdasarkan keputusan mendikbud No. 0103/0/1981 tanggal 11 maret 1981 tentang pokok-pokok kebijakan pembinaan dan pengembangan perpustaaan di Indonesia ada eberapa jenis perpustakaan antara lain:

1. Perpustakaan nasional.
2. Perpustakaan wilayah.
3. Perpustakaan umum.
4. Perpustakaan sekolah.
5. Perpustakaan perguruan tinggi.
6. Perpustakaan khusus.
7. Perpustakaan keliling.

Sejak diterbitkan kepres No.11 tahun 1989 nama jenis perpustakaan mengalami perubahan, yakni menjadi:

1. Perpustakaan nasional.
2. Perpustakaan wilayah.
3. Perpustakaan umum.
4. Perpustakaan sekolah.
5. Perpustakaan perguruan tinggi.
6. Perpustakaan khusus.
7. Perpustakaan keliling.
8. Perpustakaan tempat ibadah.

Perbedaan utama dari masing-masing jenis perpustakaan tersebut terutama pada hal tujuan, tugas dan fungsi serta masyarakat yang dilayaninya, misal tugas pokok perpustakaan nasional adalah menyelenggarakan pengembangaan, pembinaan dan pendayagunaan semua jenis perpustakaan, sedangakan

perpustakaan umum adalah untuk meningkatkan pengetahuan serta mencerdaskan masyarakat umum (mendikbud.go.id, 2017).

2.2.3 Fungsi Perpustakaan

Perpustakaan sebagai salah satu lembaga yang berperan aktif dalam peningkatan sumber informasi dan peningkatan sumberdaya alam, sangatlah penting artinya dalam usaha mencerdaskan kepentingan bangsa (Sumardji, 2001).

Perpustakaan dilambangkan sebagai tempat bertanya dalam sumber informasi tentang ilmu pengetahuan yang sifatnya khusus maupun umum. Jadi secara umum tujuan perpustakaan pada waktu sekarang ini adalah agar setiap orang yang datang ke perpustaakaan mencari informasi dan kebutuhan-kebutuhan ilmu pengetahuan tidak akan pulang dengan tangan hampa, tetapi pasti akan mendapat segala apa yang dibutuhkan (Sumardji, 2001).

Secara global perpustakaan pada umumnya mempunyai fungsi-fungsi sebagai berikut (Sumardji, 2001):

1. Sebagai sumber informasi.
2. Sebagai media dan alat pendidikan.
3. Sebagai tempat penelitian.
4. Sebagai tempat untuk kebutuhan kultur dan spiritual masyarakat.

2.2.4 Sistem Kerjasama Perpustakaan

Sebagai pusat informasi dan sumber belajar perpustakaan hendaknya mampu menyediakan informasi cepat dan mutakhir bagi para pemakainya, berbagai cara dapat dilakukan misalnya dengan membeli, meng-*copy*, meminta sumbernya dan sebagainya. Hal tersebut tidak mungkin dapat berjalan dengan baik apabila tidak ada hubungan dengan pihak lain oleh karena itu jalinan kerjasama dan komunikasi dengan lembaga atau instansi lain mutlak dilakukan, dengan demikian proses tukar menukar informasi dan kegiatan *silag laying* dapat berjalan. Selain itu kerjasama dapat juga menambah wawasan pengetahuan dan keterampilan pustakawan perguruan tinggi, lebih-lebih di era globalisasi dan informasi dewasa ini, perpustakaan akan tertinggal jauh dan tidak mau membuka atau mengembangkan cakrawala pandangan ke dunia luar.

2.2.5 Sistem Peminjaman Perpustakaan

Menurut Ningsih (2002), sistem peminjaman yang digunakan di perpustakaan ada beberapa macam, berikut beberapa sistem peminjaman dalam pustaka:

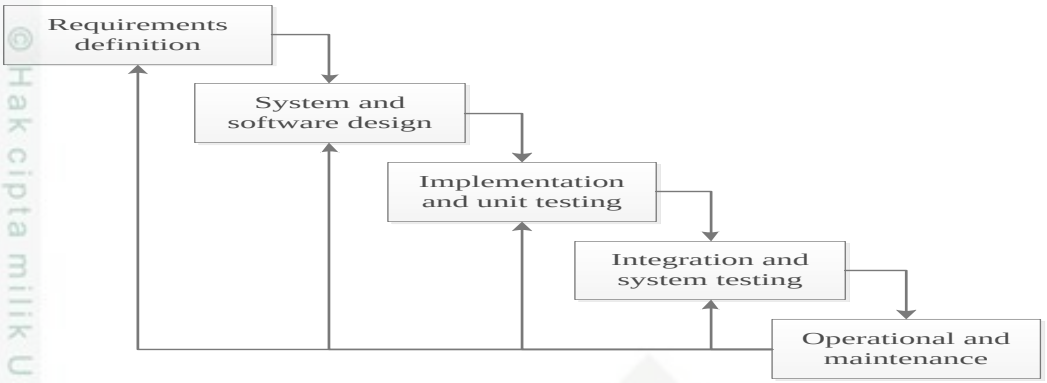
1. Sistem peminjaman menggunakan buku catatan.
   Sistem peminjaman yang menggunakan buku catatan sebagai media untuk mencatat data peminjaman.
2. Sistem peminjaman menggunakan tiket.
   Sistem peminjaman yang menggunakan tiket sebagai alat peminjaman.
3. Sistem peminjaman menggunakan formulir.
   Sistem peminjaman yang menggunakan formulir sebagai media peminjaman.
4. Sistem peminjaman menggunakan kartu pinjam.
   Sistem peminjaman yang menggunakan kartu pinjam sebagai alat peminjaman

2.3 Model Pengembangan Sistem

Model pengembangan sistem informasi yang digunakan dalam pembangunan sistem ini yaitu metode konvensional dengan memanfaatkan model atau paradigma siklus hidup klasik yang biasa disebut dengan *waterfall model* (Sommerville, 2003).

Model *waterfall* merupakan salah satu contoh dari bentuk pengembangan sistem sekuensial yang prosesnya diselesaikan setiap tahap sebelum masuk tahap berikutnya dimulai dari awal hingga akhir. Adapun kelebihan dari model ini yaitu mudah digunakan, seluruh kebutuhan sistem ini dapat didefinisikan secara utuh, eksplisit dan benar diawal proyek, maka *software engineering* (SE) dapat berjalan dengan baik dan tanpa masalah (Sommerville, 2003).

Secara umum kerangka kerja *waterfall* menurut Sommerville (2003) dapat dilihat pada Gambar 2.1.



Gambar 2.1 Kerangka Kerja Pengembangan Sistem

(Sumber: Sommerville, 2003)

Ada lima tahapan pada model *waterfall* ini, yaitu:

1. *Requirements Definition.*

Pada tahap ini dilakukan pengumpulan bahan-bahan mengenai kebutuhan-kebutuhan pengguna sistem. Setelah itu, hasil dari pengumpulan tersebut di analisa sesuai dengan apa yang diinginkan oleh pengguna. Tahapan ini dapat dilakukan dengan cara melakukan konsultasi dengan pengguna sistem. Setelah itu, didefinisikan kebutuhan-kebutuhan yang mungkin dalam sistem yang akan kita buat.

2. *System and Software Design.*

Pada proses desain sistem ini, membagi kebutuhan-kebutuhan yang telah didefinisikan pada tahap sebelumnya menjadi sistem perangkat lunak atau perangkat keras. Proses tersebut menghasilkan sebuah arsitektur sistem secara keseluruhan. Desain perangkat lunak (*software*) termasuk menghasilkan fungsi sistem perangkat lunak dalam bentuk yang memungkinkan untuk ditransformasikan kedalam satu atau lebih program yang dapat dijalankan.

3. *Implementation and Unit Testing.*

Pada tahap ini, desain perangkat lunak yang telah dihasilkan, direalisasikan kedalam bentuk program-program yang terpisah sesuai dengan unit-unitnya. Setelah terbentuk kedalam suatu program, maka dilakukan *testing* atau uji coba terhadap program tersebut.

4. *Integration and System Testing.*

Tahapan ini merupakan tahap akhir sebelum sistem diserahkan kepada pengguna. Pada tahap ini dilakukan penyatuan terhadap program-program yang telah diuji pada tahap sebelumnya. Semua program disatukan ke dalam suatu sistem yang lengkap. Setelah itu, dilakukan uji coba terakhir terhadap sistem yang telah lengkap. Setelah uji coba selesai dilakukan, maka sistem siap untuk diserahkan kepada pengguna.

5. *Operation and Maintenance*.

Pada dasarnya tahapan ini merupakan tahap yang membutuhkan waktu paling lama diantara semua tahapan. Tahapan ini merupakan tahap penggunaan sistem oleh pengguna. Pengguna akan mengetahui hasil dari sistem yang telah diinginkan. Setelah itu, dilakukan tahap perawatan atau *maintenance*. Pemeliharaan suatu software diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada permasalahan yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut.

2.4 Pendekatan Berorientasi Objek

Pendekatan berorientasi objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek.

Pada *object oriented* terdapat beberapa model pendekatan, yaitu *object oriented programming* (OOP) dan *object oriented analysis and design* (OOAD). *object oriented programming* (OOP) atau pemrograman berorientasi objek adalah konsep pemrograman yang difokuskan pada penciptaan kelas yang merupakan abstraksi atau *blueprint* atau *prototype* dari suatu objek. Kelas ini harus mengandung sifat (data) dan tingkah laku (*method*) umum yang dimiliki oleh objek-objek yang kelak akan dibuat (diinstansiasi). Data dan *method* merupakan anggota dari suatu kelas. Sedangkan *object oriented analysis and design* (OOAD) adalah metode analisis yang memeriksa *requirements* dari sudut pandang kelas

dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. beberapa model *object oriented* di atas, maka dalam peneletian ini, saya akan menggunakan model OOAD.

2.4.1 Konsep OOAD

OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu *Object Oriented Analysis* (OOA) dan OOD. OOA adalah metode analisis yang memerika *requirement* (syarat atau keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

Terdapat beberapa konsep dalam OOAD, yaitu:

1. Objek (*object*).
   a. *Object* adalah benda secara fisik dan konseptual yang ada di sekitar sebuah objek memiliki keadaan sesaat yang disebut *state*.
   b. *State* dari sebuah objek adalah kondisi dari objek atau himpunan keadaan yang menggambarkan objek tersebut. *State* dinyatakan dengan nilai dari atribut objeknya.
   c. *Atribut* adalah nilai internal suatu objek yang mencerminkan karakteristik objek, kondisi sesaat, koneksi dengan objek lain dan identitas.
   d. *Behaviour* (perilaku objek) mendefinisikan bagaimana sebuah objek bertindak dan memberi reaksi. *Behaviour* ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan oleh objek tersebut, yang dicerminkan oleh *interface*, *service* dan *method* dari objek tersebut.
   e. *Interface* adalah pintu untuk mengakses *service* dari objek.
   f. *Service* adalah fungsi yang dapat dikerjakan oleh sebuah objek.

g. *Method* adalah mekanisme internal objek yang mencerminkan perilaku objek tersebut.

2. Kelas (*class*).

Class adalah himpunan objek yang sejenis yaitu mempunyai sifat (atribut), perilaku umum (operasi), relasi umum dengan objek lain dan semantik umum. *Class* adalah abstraksi dari objek dalam dunia nyata. *Class* menetapkan spesifikasi perilaku dan atribut dari objek tersebut.

3. Kotak Hitam (*black boxes*).

Sebuah objek adalah kotak hitam, konsep ini menjadi dasar implementasi objek. Dalam operasi *object oriented* hanya *developer* yang dapat memahami *detail* proses yang ada di dalam kotak tersebut, sedangkan *user* tidak perlu mengetahui apa yang dilakukan yang penting mereka dapat menggunakan objek untuk memproses kebutuhan mereka. Kotak hitam berisi kode dan data.

a. *Encapsulation*, yaitu proses menyembunyikan *detail* implementasi sebuah objek, untuk mengakses data objek tersebut adalah melalui *interface*. Berkomunikasi dengan objek digunakan *message*.
b. *Message* adalah permintaan agar objek menerima untuk membawa metode yang ditunjukkan oleh perilaku dan mengembalikan result dari aksi tersebut kepada objek pengirim (*sender*).

4. *Asosiasi* dan *Agregasi.*

a. *Asosiasi* adalah hubungan yang mempunyai makna antara sejumlah objek. Asosiasi digambarkan dengan sebuah garis penghubung diantara objeknya. Contohnya: asosiasi karyawan dengan unit kerja. Setiap karyawan bekerja di satu unit kerja, sedangkan unit kerja dapat memiliki beberapa karyawan.
b. *Agregasi* adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada suatu objek merupakan bagian dari objek yang lain. Contohnya: kopling dan piston adalah bagian dari mesin, sedangkan mesin, roda, bodi merupakan bagian dari sebuah mobil.

2.4.2 Teknik Pemodelan dalam OOAD

Model Objek:

1. Model objek menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya.
2. Model objek berisi diagram objek. Diagram objek adalah *graph* dimana nodenya adalah kelas yang mempunyai relasi antar kelas.

Model Dinamik:

1. Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat.
2. Model dinamik dipergunakan untuk menyatakan aspek kontrol dari sistem.
3. Model dinamik berisi *state diagram*. *State diagram* adalah *graph* dimana *node*nya adalah *state* dan *arc* adalah transisi antara *state* yang disebabkan oleh *event.*

Model Fungsional:

1. Model fungsional menggambarkan transformasi nilai data di dalam sistem.
2. Model fungsional berisi data *flowdiagram*. DFD adalah suatu *graph* dimana *node*nya menyatakan proses dan *arc*nya adalah aliran data.

2.4.3 *Object Oriented Analysis* (OOA)

OOA adalah tahapan perangkat lunak dengan menentukan spesifikasi sistem (*system requirement spesification)* dan mengidentifikasi kelas-kelas serta hubungannya satu terhadap yang lain (Nugroho, 2005).

Analisa berorientasi objek (OOA) dimulai dengan menyatakan suatu masalah, analisa membuat model situasi dari dunia nyata, mengambarkan sifat yang penting. Dalam menganalisa suatu sistem analisa harus bekerja dengan pihak yang membutuhkan sistem untuk memahami masalah tersebut dengan jelas. Model analisis adalah abstraksi yang ringkas dan tepat dari apa yang harus dilakukan oleh sistem dan bagaimana melakukannya. Objek dalam model harus merupakan konsep domain dari aplikasi dan bukan merupakan implementasi komputer seperti data.

Memahami spesifikasi sistem, kita perlu mengidentifikasi para pengguna atau yang sering disebut sebagai aktor-aktor. Siapa aktor-aktor yang akan menggunakan sistem dan bagaimana mereka menggunaka sistem.

Aktivitas utama dari OOA adalah:

a. Menganalisis masalah *domain*.
b. Menjelaskan sistem proses.
c. Mengidentifikasi objek.
d. Menentukan atribut.
e. Mendifinisikan operasi.
f. Komunikasi antar objek.

2.4.4 Object Oriented Design (OOD)

Desain berorientasi objek (OOD) merupakan tahap lanjutan setelah analisis berorientasi objek, dimana tujuan sistem diorganisasi ke dalam subsistem berdasarkan struktur analisis dan arsitektur yang dibutuhkan. Desainer sistem (*System Desginer*) menentukan karateristik penampilan secara optimal, menentukan strategi memecahkan masalah dan menentukan pilihan alokasi sumber daya.

Desain model yang digunakan berdasarkan model analisis dengan dilengkapi rincian untuk implementasi. Fokus dari desain objek (*object design*) adalah perencanaan struktur data dan algoritma yang diperlukan untuk implementasi setiap kelas. Objek *domain* komputer dijelaskan dengan menggunakan konsep dan notasi berorientasi objek yang sama.

OOD adalah merancang kelas-kelas yang teridentifikasi selama tahap analisis dan antarmuka (*user interface*). Selama tahap ini, kita mengidentifikasi dan menambahkan beberapa objek dan kelas yang mendukung implementasi dari spesifikasi kebutuhan (Nugroho, 2005).

Proses OOD:

1. Mendefiniskan kontek dan mode dari penggunaan sistem.
2. Mendesain arsitrektur sistem.
3. Indentifikasi objek sistem utama.

4. Mengembangkan model desain.
5. Menentukan *interface* objek.

2.4.5 Karateristik dari Objek

a. Objek

1. Identitas berarti bahwa data diukur mempunyai nilai tertentu yang membedakan entitas disebut objek.
2. Objek dapat kongkrit, seperti halnya arsip dalam sistem, atau konseptual seperti kebijakan penjadwalan dalam *multiprocessing* pada sistem operasi.
3. Setiap objek mempunyai sifat yang melekat pada identitasnya.
4. Dua objek dapat berbeda walaupun bila semua nilai atributnya identik.

b. Kelas Objek

1. Kelas merupakan gambaran sekumpulan objek yang terbagi dalam atribut, operasi, metode, hubungan dan makna yang sama.
2. Suatu kegiatan mengumpulkan data (*atribut*) dan perilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup.
3. Kelas objek merupakan wadah bagi objek. Dapat digunakan untuk menciptakan objek.
4. Objek mewakili fakta atau keterangan dari sebuah kelas.

2.4.6 Karakteritik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama, yaitu:

1. *Encapsulation* (Pengkapsulan).

Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

2. *Inheritance* (*Pewarisan*).

Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan

metode dari objek induk diturunkan kepada anak objek, demikian seterusnya, karena telah mewarisi sifat induknya. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiliki bersama diantara kelas yang mempunyai hubungan secara hirarki. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya dan ditambah dengan sifat unik yang dimilikinya, kelas objek dapat didefinisikan atribut dan *service* dari kelas objek lainnya. *Inheritance* menggambarkan generalisasi sebuah kelas.

Sifat yang dimiliknya oleh kelas induknya tidak perlu diulang dalam setiap subkelas. Sebagai contoh, mobil dan sepeda motor adalah subkelas dari kendaraan bermotor. Kedua subkelas mewarisi sifat yang dimiliki oleh kendaraan bermotor, yaitu (Nugroho, 2005):

1. Mempunyai mesin.
2. Dapat berjalan.

Kedua subkelas mempunyai sifat masing-masing yang berbeda, misalnya jumlah roda dan kemampuan untuk berjalan mundur yang tidak dimiliki oleh sepeda motor. Beberapa faktor dari superkelas yang bersifat umum dan dimasukkan ke dalam kelas induknya serta mewariskan sifat tersebut pada kelas turunannya, sehingga mengurangi pengulangan yang terjadi dalam desain dan pemrograman. Hal ini merupakan keuntungan dari sistem berorientasi objek.

3. *Polymorphism* (Perbedaan Bentuk).

Polimorfisme yaitu konsep yang menyatakan bahwa seuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. *Polimorfisme* mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon *message* yang sama. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan objek.

2.5 Model Umum Perancangan Analisis dan Perancangan Sistem

Adapun model perancangan analisis dan perancangan sistem yang akan digunakan adalah:

2.5.1 *Unified Modelling Language* (UML)

UML adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *object oriented.* UML juga merupakan sistem notasi yang membantu pemodelan sistem menggunakan konsep berorientasi objek (Nugroho, 2005).

2.5.2 Definisi Umum *Unified Modelling Language* (UML)

UML adalah sebuah alat bantu yang sangat handal di dunia pengembangan sistem berorientasi objek, hal ini disebabkan karena UML menyediakan bahasa pemodelan *visual* yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain (Nugroho, 2005).

UML adalah bahasa *universal* untuk memvisualisasikan grafis model yang tepat, menetapkan model yang tepat, lengkap dan tidak ambigu untuk mengambil semua keputusan penting dalam analisis, desain dan implementasi, membangun model yang dapat dihubungkan langsung dengan bahasa pemrograman, mendokumentasikan semua informasi yang dikumpulkan oleh tim sehingga memungkinkan untuk berbagi informasi.

Dalam proyek pengembangan sistem apapun, fokus utama dalam analisis dan perancangan adalah model. Menurut Nugroho (2005), dengan model kita dapat merepresentasikan sesuatu karena:

a. Model mudah dan cepat untuk dibuat.
b. Model bisa digunakan sebagai simulasi untuk mempelajari lebih detail tentang sesuatu.
c. Model bisa dikembangkan sejalan dengan pemahaman kita.
d. Model bisa mewakili sesuatu yang nyata maupun tidak nyata.

2.5.3 Diagram-Diagram *Unified Modelling Language* (UML)

UML mempunyai sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. Karena ini merupakan sebuah bahasa, UML mempunyai aturan untuk menggabungkan dan mengkombinasikan elemen-elemen tersebut.

Dalam membangun suatu model perangkat lunak dengan UML, digunakan bentuk-bentuk diagram atau simbol untuk merepresentasikan elemen-elemen dalam sistem.

Bentuk diagram yang digunakan untuk merepresentasikannya adalah sebagai berikut (Nogroho, 2005):

a. *Use Case diagram.*
b. *Activity diagram.*
c. *Sequence diagram.*
d. *Class diagram.*
e. *Collaboration diagram.*
f. *Statechart diagram.*
g. *Component diagram.*
h. *Deployment diagram.*

Tabel 2.1 Tipe Diagram UML

Diagram	**Tujuan**
Use Case	Menunjukkan sekumpulan kasus fungsional dan aktor dan hubungannya.
Activity	Pandangan operasi, bagaimana objek-objek bekerja, aksi-aksi yang mempengaruhi objek, pandangan *use case workflow*.
Sequence	Berfungsi untuk *overview* prilaku sistem, menunjukkan objek-objek yang diperlukan, mendokumentasikan skenario dari suatu diagram *Use Case*, memeriksa jalur-jalur pengaksesan.
Class	Memodelkan kosakata di sistem, distribusi dan tanggung jawab, tipe primitif, kolaborasi, skema *database* logik.
Collaboration	Memodelkan pandangan prilaku sistem pada *link-link* di antara objek-objek. Ilustrasi dari *use case*, memeriksa jalur-jalur pengaksesan.

Tabel 2.1 Tipe Diagram UML (Lanjutan)

Diagram	Tujuan
Statechart	Pandangan objek secara waktu, pandangan dalam berkaitan dengan rangsangan eksternal.
Component	Memodelkan *file* yang dapat dieksekusi dan pustaka, memodelkan tabel, *file* dan dokumen, memodelkan API (*Application Programming Interupt*).
Deployment	Konfigurasi pemprosesan saat jalan dan komponen-komponen yang terdapat di dalamnya.

2.5.4 Diagram-Diagram UML yang Digunakan

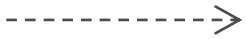
1. *Use Case Diagram*

Diagram *use case* merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor dan hubungannya. Diagram *use case* adalah penting untuk memvisualisasikan, memspesifikasikan dan mendokumentasikan kebutuhan perilaku sistem. Diagram *use case* merupakan pusat pemodelan perilaku sistem, subsistem, kelas. Berikut adalah elemen dalam *use case.*

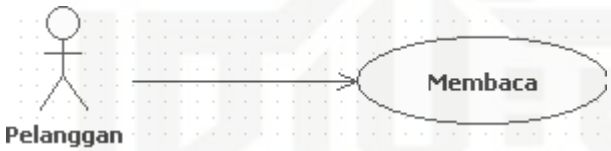
Tabel 2.2 Notasi *Use Case Diagram*

Penjelasan	Notasi UML
Actor: Mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*.	
Use case: Abstraksi dari interaksi antara sistem dan *actor.*	
Association: adalah abstraksi dari penghubung antara *actor* dan *use case.*	

Tabel 2.2 Notasi *Use Case Diagram* (Lanjutan)

Penjelasan	Notasi UML
Generalisasi: menunjukkan spesialisasi *actor* untuk dapat berpartisipasi dalam *use case.*	

Pelanggan datang melakukan pencarian buku untuk dibaca, dengan cara melihat dan membaca buku yang tersedia untuk dibaca sesuai dengan selera.



Gambar 2.4 *Use Case Diagram*

Gambar tersebut memberikan pemahaman bahwa pelanggan melakukan proses ‘membaca’, proses yang ada di *use case* ini juga dapat mendeskripsikan bahwa ‘objek’ (buku, informasi, data) dapat dibaca oleh pelanggan (*actor*).

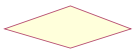
2. *Activity Diagram*

Pada dasarnya, diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Kegunaan diagram ini adalah untuk memodelkan *workflow* atau jalur kerja, memodelkan operasi, bagaimana objek-objek bekerja, aksi-aksi dan pengaruh terhadap objek. Simbol-simbol yang terdapat dalam *activity diagram*, sebagai berikut:

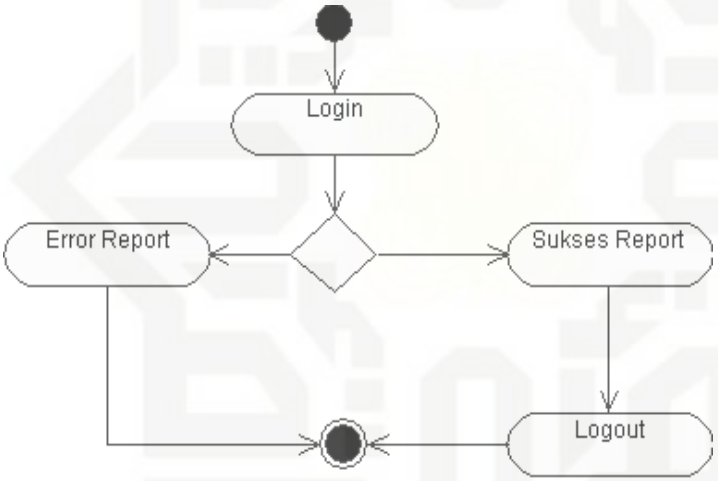
Tabel 2.3 Simbol *Activity Diagram*

Keterangan	Simbol
Titik awal atau permulaan.	
Titik akhir atau akhir dari aktivitas.	
Activity atau aktivitas yang dilakukan oleh aktor.	

Tabel 2.3 Simbol *Activity Diagram* (Lanjutan)

Keterangan	Simbol
Decision atau pilihan untuk mengambil keputusan.	
Arah tanda panah alur proses.	

Activity diagram merupakan salah satu diagram yang umum digunakan dalam UML untuk menjabarkan proses atau aktivitas dari aktor. Sebagai contoh, pelanggan melakukan *login* (masuk) pada halaman *website* untuk bergabung, jika pelanggan belum terdaftar, maka akan ditolak oleh sistem dan dikembalikan, proses penjabarannya adalah sebagai berikut:



Gambar 2.5 *Activity Diagram*

Dalam *activity diagram* tersebut dijelaskan bahwa *user* melakukan proses *login* untuk dapat memasuki area sistem, jika proses *login* dan atau *user* belum teregistrasi, maka *user* akan ditolak oleh sistem tersebut dan diberi pesan *error*. Selain itu, bila *user* telah teregistrasi dan memasukkan kode *login* dengan benar maka akan diberi akses untuk masuk ke sistem dan diberikan pesan sukses. *User* dapat *logout* (keluar) untuk mengakhiri sesi.

3. *Sequence Diagram*

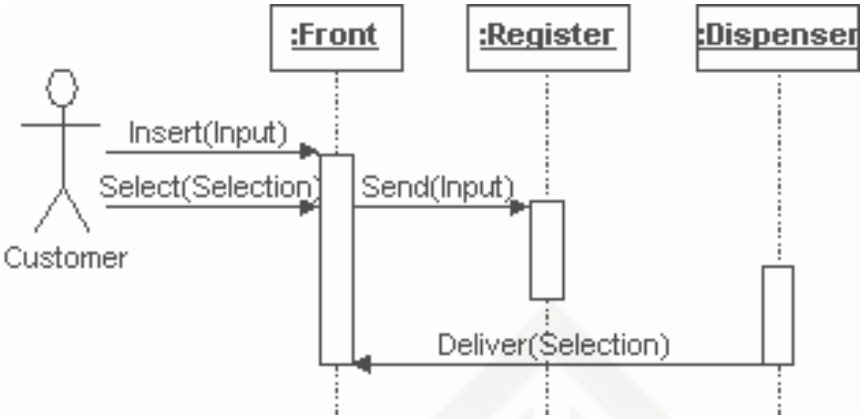
Sequence diagram mendokumentasikan komunikasi atau interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah objek dan *message* (pesan) yang

diletakkan diantara objek di dalam *use case*. Perlu diingat bahwa di dalam diagram ini, kelas-kelas dan aktor-aktor diletakkan dibagian atas diagram dengan urutan dari kiri ke kanan dengan garis *lifeline* yang diletakkan secara vertikal terhadap kelas dan aktor. Berikut adalah notasi-notasinya:

Tabel 2.4 Notasi pada *Sequence Diagram*

Nama	Keterangan	Objek
Object	*Object* merupakan *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal.Digambarkan sebagai sebuah *class* (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma.	: *Object*1
Actor	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom. Simbol a*ct*or sama dengan simbol pada *actor use case diagram*.	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah objek.	
Activation	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline.* *Activation* mengindikasikan sebuah objek yang akan melakukan sebuah aksi.	
Massage	*Message* digambarkan dengan anak panah horizontal antara *activation*. *Message* mengindikasikan komunikasi antara *object-object.*	*Message* 

Berikut adalah contoh sebuah *sequence diagram*:



Gambar 2.6 *Sequence Diagram*

4. *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi obyek. *Class* menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode atau fungsi). Berikut adalah notasi yang ada pada *class diagram*:

Tabel 2.5 Notasi pada *Class Diagram*

Nama	Keterangan	Objek
Class	*Class* adalah blok-blok pembangun pada pemrograman berorientasi objek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas tiga bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan properti atau atribut *class*. Bagian akhir mendefinisikan *method* dari sebuah *class*.	Site Config +sqlDNS:string +Adminemail:String
Assosiation	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara dua *class* dan dilambangkan oleh sebuah garis yang menghubungkan.	1..n Owned by 1

5. *Collaboration Diagram*

Collaboration diagram menggunakan prinsip yang sama dengan *sequence diagram*, sama-sama memodelkan interaksi antar objek-objek, yang membedakannya hanya cara penggambarannya saja. Pada *collaboration diagram* ini, objek-objek dan *message* (pesan) yang ada digambarkan mirip seperti flowchart, hanya saja, untuk menjaga urutan pesan yang diterima oleh masing-masing objek, pesan-pesan tersebut diberi nomor urutan pesan. Berikut adalah notasi untuk *collaboration diagram*:

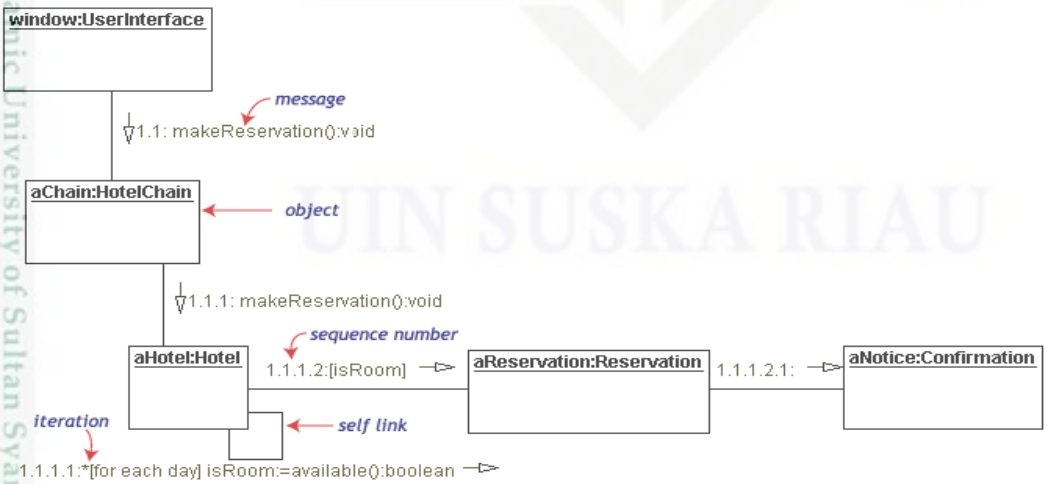
Tabel 2.6 Notasi pada *Collaboration Diagram*

Nama	Keterangan	Objek
Object	*Object* merupakan *instance* dari sebuah *class*. Digambarkan sebagai sebuah c*lass* (kotak) dengan nama objek di dalamnya yang diawali dengan sebuah titik koma.	: Object1
Actor	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat disertakan ke dalam *collaboration diagram*. Simbol *actor* sama dengan simbol pada *actor use case diagram*.	
Massage	*Message*, digambarkan dengan anak panah yang mengarah antar objek dan diberi label urutan nomor yang mengindikasikan urutan komunikasi yang terjadi antar objek.	**Massage** →
Assosiation	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara dua *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara dua *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship* (Contoh: *One-to-one, one-to-many, many-to-many*).	1..n Ownedby1

Tabel 2.6 Notasi pada *Collaboration Diagram* (Lanjutan)

Nama	Keterangan	Objek
Composition	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.	
Dependency	Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.	

Berikut adalah contoh dari *collaboration diagram*:

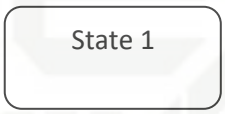


Gambar 2.7 *Collaboration Diagram*

6. *Statechart Diagram*

Statechart diagram atau yang biasa juga disebut *state diagram* digunakan untuk mendokumentasikan beragam kondisi atau keadaan yang bisa terjadi terhadap sebuah *class* dan kegiatan apa saja yang dapat merubah kondisi atau keadaan tersebut. Contohnya sebuah televisi yang dapat berada dalam kondisi menyala atau mati, jika tombol *"power"* ditekan maka televisi akan menyala, begitu juga sebaliknya akan mati jika tombol *"power"* ditekan kembali. Maka disini kita mempunyai sebuah kelas yaitu televisi, dua *state* yaitu menyala dan mati dan dua *transition* yaitu menyalakan TV dan mematikan TV. Tidak seperti diagram-diagram *behavioural* lainnya yang memodelkan interaksi diantara beberapa *class*, *state diagram* justru biasanya hanya memodelkan transisi yang terjadi hanya pada sebuah *class*. Berikut adalah notasi *statechart diagram*:

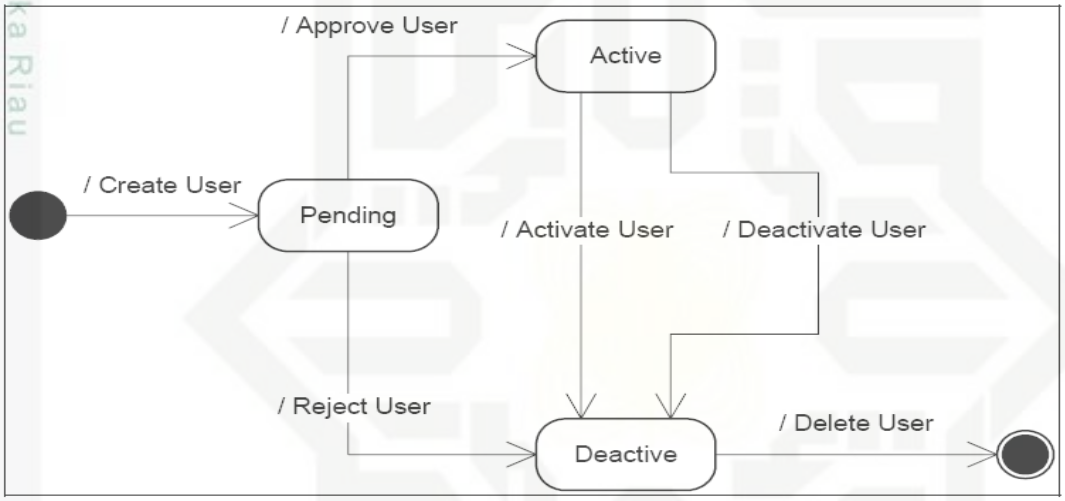
Tabel 2.7 Notasi pada *Statechart Diagram*

Nama	Keterangan	Objek
State	Notasi *state* menggambarkan kondisi sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya tumpul dengan nama *state* di dalamnya.	
Transition	Sebuah *transition* menggambarkan sebuah perubahan kondisi objek yang disebabkan oleh sebuah *event*. *Transition* digambarkan dengan sebuah anak panah dengan nama *event* yang ditulis diatasnya, dibawahnya atau sepanjang anak panah tersebut.	
Initial state	*Initial state* adalah sebuah kondisi awal sebuah *object* sebelum ada perubahan keadaan. *Initial state* digambarkan dengan sebuah lingkaran solid. Hanya satu *initial state* yang diizinkan dalam sebuah diagram.	

Tabel 2.7 Notasi pada *Statechart Diagram* (Lanjutan)

Nama	Keterangan	Objek
Final State	*Final state* menggambarkan ketika objek berhenti memberi respon terhadap sebuah *event*. *Final state* digambarkan dengan lingkaran solid didalam sebuah lingkaran kosong.	

Berikut adalah contoh sebuah *statechart diagram*:



Gambar 2.8 *Statechart Diagram*

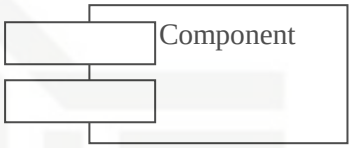
7. *Component Diagram*

Komponen perangkat lunak adalah bagian fisik dari sebuah sistem yang menetap di komputer. Komponen merupakan implementasi *software* dari sebuah *class*. Komponen biasa berupa tabel, *file* .data, *file* .exe, *file* .DLL, dokumen dan lain-lain.

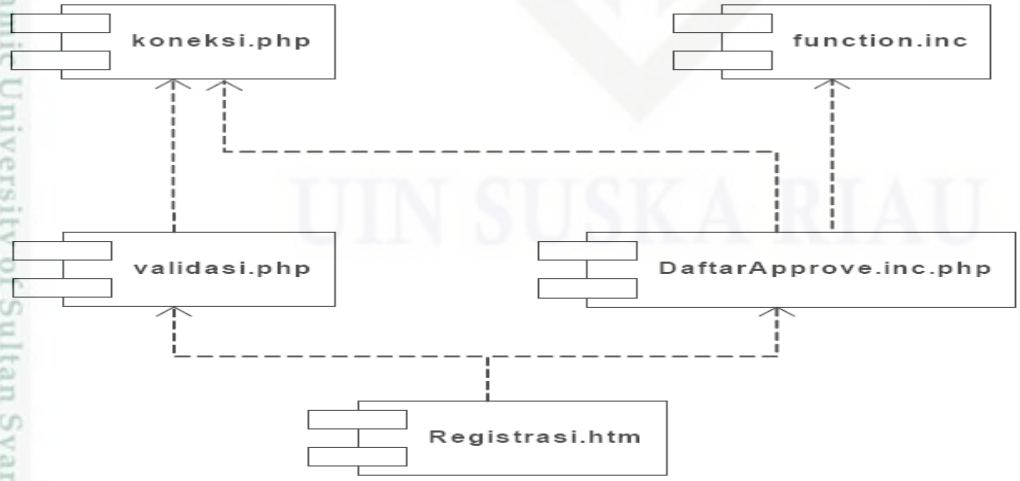
Component diagram mengandung komponen *interface* dan *relationship*. Komponen diagram ini digunakan pada saat anda ingin memecah sistem menjadi komponen-komponen dan ingin menampilkan hubungan-hubungan mereka dengan antarmuka atau pemecahan komponen menjadi struktur yang lebih rendah. Secara umum dapat kita katakan bahwa komponen diagram digunakan untuk menjelaskan kebergantungan antar beragam komponen-komponen *software*

seperti misalnya kebergantungan antara *file-file executable* dengan *file-file* sumbernya (*source file*) dan lain-lain. Berikut adalah notasi dari *component diagram*:

Tabel 2.8 Notasi pada *Component Diagram*

Nama	Keterangan	Objek
Component	Sebuah komponen melambangkan sebuah entitas *software* dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.	
Depedency	Sebuah *dependency* digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempat sebuah komponen itu bergantung.	

Berikut adalah contoh sebuah *component* diagram:

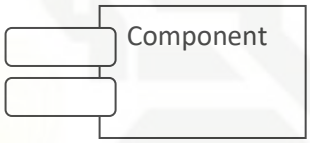
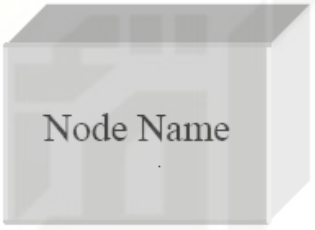


Gambar 2.9 *Component Diagram*

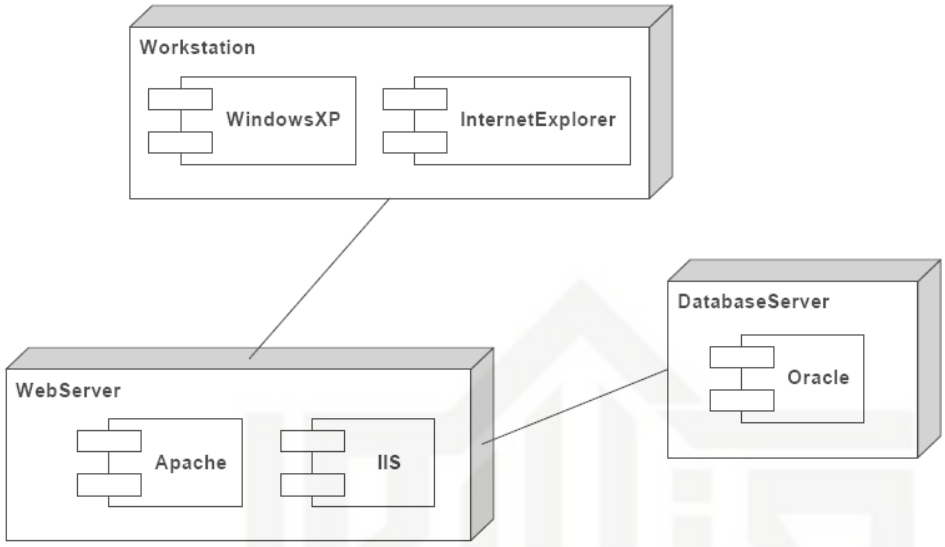
8. *Deployment Diagram*

Deployment diagram menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware* yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen *hardware* tersebut. *Deployment diagram* dapat digunakan pada bagian-bagian awal proses perancangan sistem untuk mendokumentasikan arsitektur fisik sebuah sistem. Berikut adalah notasi-notasi yang digunakan pada *deployment diagram*:

Tabel 2.9 Notasi pada *Deployment Diagram*

Nama	Keterangan	Objek
Component	Pada *deployment diagram*, komponen komponen yang ada diletakkan di dalam *node* untuk memastikan keberadaan posisi mereka.	
Node	*Node* menggambarkan bagian-bagian *hardware* dalam sebuah sistem. Notasi untuk node digambarkan sebagai sebuah kubus tiga dimensi.	
Association	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara *element-element hardware*.	

Berikut adalah contoh sebuah *deployment diagram*:



Gambar 2.10 *Deployment Diagram*

2.6 Konsep Dasar Perancangan *Database*

Pada dasarnya perancangan sistem *database* berguna untuk menghasilkan informasi. Informasi yang tepat dan akurat membantu manejer dalam pengambilan keputusan dan menentukan langkah-langkah yang harus diambil untuk mempertahankan dan mengembangkan organisasi dan usaha. Informasi juga mendukung kegiatan operasional dan manajerial organisasi atau perusahaan. Semua itu dapat diperoleh dengan suatu sistem penyediaan basisdata yang lengkap, akurat dan dapat ditampilkan secara tepat dan mudah setiap kali diperlukan.

2.6.1 Defenisi *Database*

Database adalah kumpulan *file* yang saling berelasi, relasi tersebut biasa ditujukan dengan kunci dari tiap *file* yang ada. Suatu *database* menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instasi (Kristanto, 2008).

Dua tujuan utama dari konsep *database* adalah kemapuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan data program yang memproses data. Indenpendensi data dicapai dengan menempatkan spesifikasi

dalam tabel dan kamus yang terpisah secara fisik dari program. Defenisi *entity*, *attribute*, data *value*, *record*, *file*, DBMS (Kristanto, 2008):

a. *Entity.*

   Entity adalah orang, tempat, kejadian atau konsep yang informasinya direkam.

b. *Attribute.*

   Setiap *entity* mempunyai *attribute* atau sebutan untuk mewakili suatu entity. *Attribute* juga disebut sebagai data elemen, data *field*, data *item*.

c. Data *Value.*

   Data *value* adalah data *actual* atau informasi yang disimpan pada setiap elemen atau *attribute*.

d. *Record.*

   Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu *entity* secara lengkap.

e. *File.*

   Kumpulan *record*-record sejenis mempunyai panjang elemen yang sama, *attribute* yang sama, namun berbeda-beda *value*-nya.

f. *Database.*

   Sekumpulan data yang terintegrasi yang diorganisasi untuk memenuhi kebutuhan para pemakai di dalam suatu organisasi.

g. *Database Management System* (DBMS).

   DBMS adalah kumpulan *file* yang saling berkaitan bersama dengan program untuk pengolahannya.

2.6.2 Langkah-langkah dalam Mendesain *Database*

Agar *database* yang dirancang dapat lebih mudah maka diperlukan langkah-langkah mendesainnya. Langkah-langkah mendesain *database* (Kristanto, 2008):

1. *Conceptual Model.*

   Adalah mendefenisikan data-data yang diperlukan pada langkah pertama ini yang harus dipikirkan adalah data apa saja yang diinginkan sebagai

output, baik *input* melalui *layer* atau *printer* yang data-datanya harus disimpan ke dalam suatu *file database*.

2. *Logical Database Design.*

   Adalah menentukan data-data yang akan dikelompokkan dalam suatu *file database*, hal yang penting dalam pengembangan *logic desaign database* model relasi adalah proses nomalisasi, yaitu pengelompokan data-data menjadi suatu tabel berdasarkan *entity* dan relasinya.

3. *Pysical Database Design*.

   Adalah mempertimbangkan kemampuan sistem yang akan dipakai, jika perlu melakukan perubahan sehingga dengan informasi sistem.

2.6.3 Rancangan *Database*

Merancang *database* merupakan suatu hal yang sangat penting, kesulitan utama dalam merancang *database* adalah bagaimana merancang suatu *database* yang dapat dipakai pada saat ini dan mendatang.

Dalam merancang *database* konsep yang harus dipahami terlebih dahulu yaitu (Kristanto, 2008):

1. *Candidate key* (kunci kandidat atau kunci calon).

   Adalah satu *attribute* yang mengidentifikasi secara unik suatu kejadian spesifik dari entity.

   Satu minimal set dari *attribute* menyatakan secara tak langsung di mana anda tidak dapat membuang beberapa *attribute* dalam set tanpa memasukkan kepemilikan yang unik.

   Jika satu kunci *candidate* berisi lebih dari satu *attribute*, maka biasanya disebut sebagai *composite key* (kunci campuran atau gabungan).

2. *Primary key* (kunci primer)

   Adalah satu *attribute* atau satu set minimal *attribute* yang tidak hanya mengindentifikasi secara unik suatu kejadian spesifik, tapi juga dapat mewakili setiap kejadian dari suatu *entity*.

   Setiap kunci kandidat punya peluang menjadi *primary key*, tetapi sebaliknya dipilih satu saja yang dapat mewakili secara menyeluruh terhadap *entity* yang ada.

3. *Alternate key* (kunci *alternative*).

   Adalah kunci kandidat yang tidak dipakai sebagai *primary key*. Kerap kali kunci *alternative* dipakai sebagai kunci pengurutan dalam laporan.

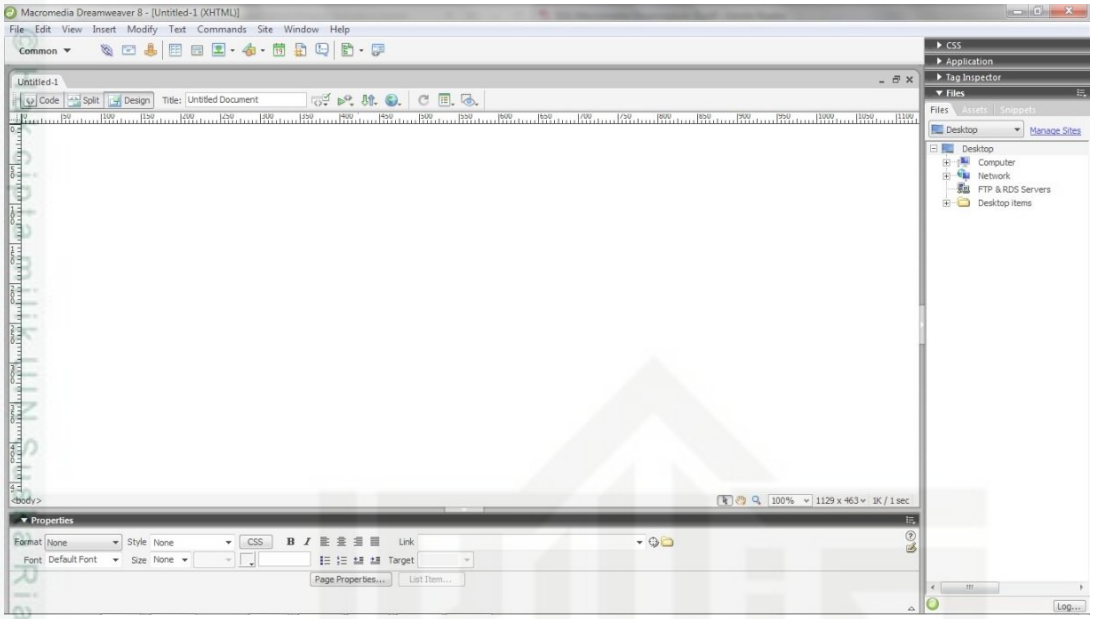
   Salah satu alat bantu yang dapat menggambarkan rancangan suatu *database* adalah ER_Model. ER_Model adalah alat bantu dalam perancangan *database*. Dalam perancangan konsep data digunakan model data relational. Model data relational merupakan suatu model untuk menjelaskan hubungan antar data dalam suatu *database*.

2.7 Pengenalan Macromedia Dreamweaver

Macromedia Dreamweaver adalah program untuk membuat dan meng-*edit* dokumen HTML secara *visual* dan mengelola halaman sebuah situs. Dreamweaver menyediakan banyak perangkat yang berkaitan dengan pengkodean dan fitur seperti HTML, CSS, JavaScript, PHP, ASP, ColdFusion, dan XML (Romzi, 2004).

Dreamweaver 8 mempunyai kehandalan dalam membuat dan mendesain *web* tanpa harus menuliskan *tag* HTML satu persatu. Dreamweaver 8 menggunakan metode klik dan *drag* yang dapat mempermudah anda dalam membuat *website* dengan cepat, mudah, menarik dan interaktif. Dreamweaver 8 juga mempunyai kemampuan untuk mendukung pemrograman *server side* dan *client side*. *Server side* digunakan untuk memproses data yang berhubungan dengan *sever*, misal pengolahan *database*. *Client side* merupakan bahasa pemrograman tambahan sekaligus sebagai pelengkap dari bahasa pemrograman lainnya (Hadi, 2006).

Bagian-bagian jendela pada macromedia dreamweaver 8 adalah sebagai berikut (Hadi, 2006):



Gambar 4.2 Macromedia Dreamweaver

1. *Insert Bar* digunakan untuk memasukkan atau membuat berbagai macam objek ke dalam halaman *web* yang sedang dibuat pada *document window*.
2. Menu *Bar* berisi berbagai macam menu untuk mengatur halaman *web* yang sedang dibuat. Misalnya menu *file*, *edit*, *view* dan lain-lain.
3. *Property Inspector* berisi berbagai macam atribut dari elemen yang sedang terpilih dalam *document window*. Misalkan anda sedang mengklik sebuah tabel pada bagian *document window*, maka *property inspector* akan menampilkan berbagai macam atribut dari tabel tersebut. Anda juga dapat mengubah nilai-nilai atributnya pada bagian ini.
4. *Property Inspector* akan menampilkan berbagai macam atribut dari tabel tersebut. Anda juga dapat mengubah nilai-nilai atributnya pada bagian ini.
5. Panel *Groups* berisi berbagai macam panel. Setiap panel digunakan untuk mengatur hal-hal yang spesifik. Misalkan panel CSS digunakan untuk mengatur CSS. Panel *Files* digunakan untuk mengatur *file-file* dan lain sebagainya. Klik tombol panah kecil di pojok kiri atas sebuah *panel* untuk membuka atau menutup panel tersebut.

2.8 Pengenalan Xampp

XAMPP merupakan *tool* yang menyediakan paket perangkat lunak ke dalam satu buah paket. 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 (Taryana, 2009).

Software XAMPP terdiri atas:

a. Apache versi 2.0.54.
b. MySQL versi 4.1.12.
c. PHP versi 5.0.4.
d. phpMyAdmin versi 2.6.2-p11.

2.8.1 Mengenal APACHE

Apache sudah berkembang sejak versi pertamanya, sampai saat ditulisnya artikel ini versi terakhirnya yang ada yaitu Apache ver 2.0.54. Apache bersifat *open source*, artinya setiap orang boleh menggunakannya, mengambil dan bahkan mengubah kode programnya (Taryana, 2009).

Tugas utama apache adalah menghasilkan halaman *web* yang benar kepada peminta, berdasarkan kode PHP yang dituliskan oleh pembuat halaman web. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam MySQL) untuk mendukung halaman *web* yang dihasilkan (Taryana, 2009).

2.8.2 Mengenai PHP

Bahasa pemrograman PHP merupakan bahasa pemrograman untuk mebuat *web* yang bersifat server-side *scripting*. PHP memungkinkan kita untuk membuat halaman *web* yang bersifat dinamis. PHP dapat dijalankan pada berbagai macam *operating system* (OS), misalnya Windows, Linux dan Mac OS. Selain Apache, PHP juga mendukung beberapa *web server* lain, misalnya Microsoft IIS, Caudium, PWS dan lain-lain (Evangelos, 2002).

Seperti penjelasan sebelumnya bahwa PHP dapat memanfaatkan *database* untuk menghasilkan halaman *web* yang dinamis. Sistem manajemen *database* yang sering digunakan bersama PHP adalah MySQL. Namun PHP juga

mendukung sistem manajemen *Database Oracle, Microsoft Acces, Interbase*, d-Base, PostgreSQL dan sebagainya (Evangelos, 2002).

Hingga kini PHP sudah berkembang hingga versi ke 5. PHP 5 mendukung penuh *object oriented programing* (OOP), integrasi XML, mendukung semua ekstensi terbaru MySQL, pengembangan *web services* dengan SOAP dan REST, serta ratusan peningkatan kemampuan lainnya dibandingkan versi sebelumnya. Sama dengan *web server* lainnya PHP juga bersifat *open source* sehingga setiap orang dapat menggunakannya dengan gratis (Evangelos, 2002).

2.8.3 Mengenai MySQL

Perkembangannya disebut SQL yang merupakan kepanjangan dari *structured query language*. SQL merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. SQL pertama kali didefinisikan oleh *American National Standards Institute* (ANSI) pada tahun 1986. MySQL adalah sebuah sistem manajemen *database* yang bersifat *open source*. MySQL adalah pasangan serasi dari PHP. MySQL dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia (Wahana, 2011).

MySQL dapat digunakan untuk membuat dan mengola *database* beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah dan menghapus data yang berada dalam *database*. MySQL merupakan sistem manajemen *database* yang bersifat *at relational*. Artinya data-data yang dikelola dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat (Wahana, 2011).

MySQL dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar. MySQL juga dapat menjalankan perintah-perintah *Structured Query Language* (SQL) untuk mengelola *database-database* yang ada di dalamnya. Hingga kini, MySQL sudah berkembang hingga versi 5. MySQL 5 sudah mendukung *trigger* untuk memudahkan pengelolaan tabel dalam *database* (Wahana, 2011).

2.8.4 Mengenai PHPMyAdmin

Pengelolaan *database* dengan MYSQL harus dilakukan dengan mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud

tertentu. Jika anda ingin membuat *database*, ketikkan baris perintah yang sesuai untuk membuat *database*. Jika kita ingin menghapus tabel, ketikkan baris perintah yang sesuai untuk menghapus tabel. Hal tersebut tentu cukup menyulitkan karena kita harus hafal dan mengetikkan perintahnya satu persatu (Evangelos, 2002).

Banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola database dalam MySQL, salah satunya adalah phpMyAdmin. PhpMyAdmin dapat membuat tabel, mengisi data dan lain-lain dengan mudah tanpa harus hafal perintahnya. Mengaktifkan phpMyAdmin langkah-langkahnya adalah: yang pertama setelah XAMPP kita terinstall, kita harus mengaktifkan *web server* Apache dan MySQL dari *control* panel XAMPP. Yang kedua, jalankan *browser* kesayangan anda (IE, Mozilla Firefox atau Opera) lalu ketikkan alamat *web* berikut: http://localhost/phpmyadmin/ pada *address bar* lalu tekan *enter*. Langkah ketiga apabila telah nampak *interface* (tampilan antar muka) phpMyAdmin anda bisa memulainya dengan mengetikkan nama *database*, nama tabel dan seterusnya (Evangelos, 2002).