**BAB II**

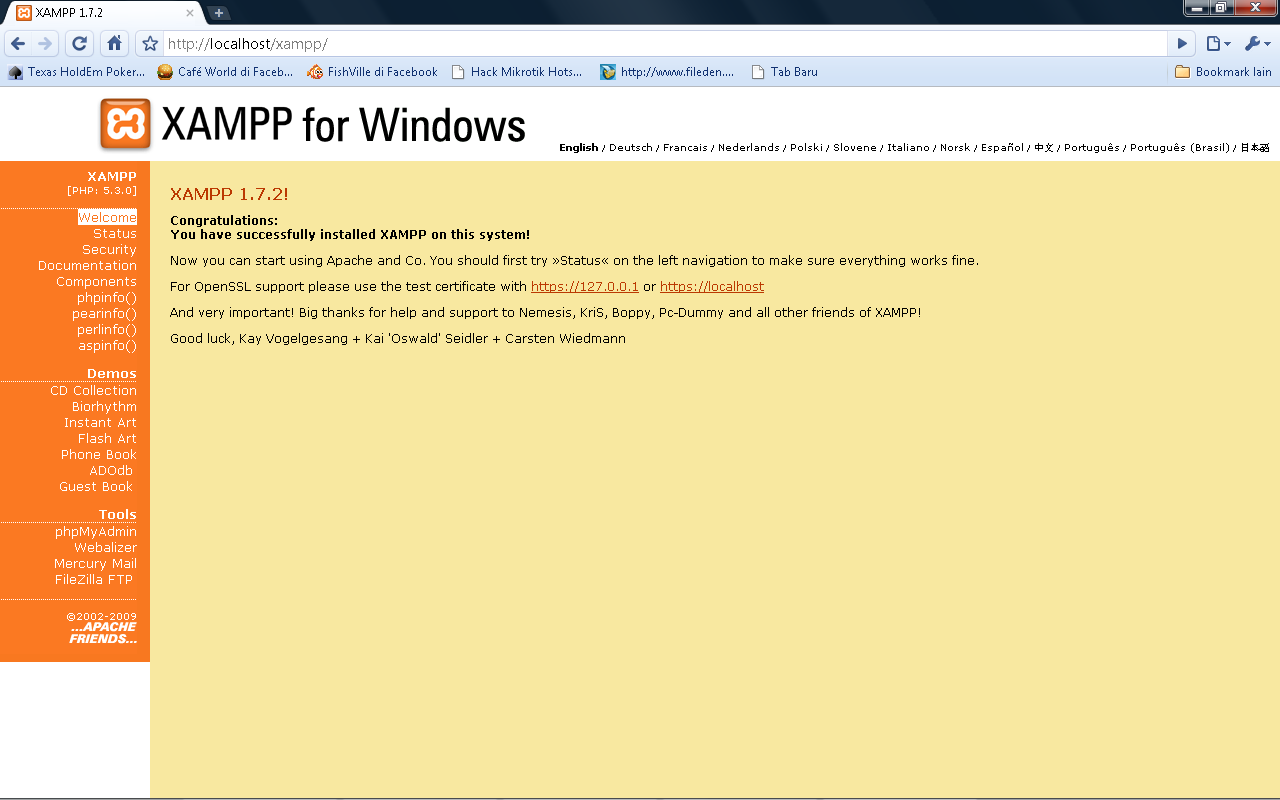
**LANDASAN TEORI**

**2.1. *PHP***

*PHP Hypertext Preprocessor (PHP)* merupakan bahasa yang mampu menghasilkan aplikasi *web* dinamis. Sejak dikembangkannya *PHP* pada awal 1995 oleh Ramus Lerdorf, *PHP* menjadi pilihan bahasa pemrograman web yang banyak digunakan oleh situs terkemuka. Sebutlah [www.friendster.com](http://www.friendster.com), pada tahun 2002 beralih ke *PHP* setelah sebelumnya menggunakan *ASP*, ini dikarenakan *PHP* mempunyai keunggulan dibanding bahasa pemrograman yang lain, diantaranya: (Prasetyo, Dwi, 2005)

1. *PHP* adalah bahasa pemrograman *open source* dengan lisensi *GPL*, sehingga semua orang dapat menggunakannya untuk keperluan komersil ataupun non-komersil tanpa dipungut biaya.
2. *PHP* gratis dan dapat dijalankan pada semua *web server* dan sistem operasi.
3. Kemampuan *PHP* untuk berhubungan dengan berbagai database terkemuka.
4. *Content type modification*, *feature* ini dapat menghasilkan bukan hanya file *hypertext* saja, tetapi dapat menghasilkan gambar, file *PDF*, *Movie Flash*, ataupun *Page Redirecting*.

Keunggulan *PHP* tersebut masih banyak lagi, karena bersifat *open source,* *PHP* sudah dikembangkan ke berbagai keperluan lain, seperti GIS (MapServer), .net complient (untuk *PHP* versi 5 keatas). Berikut adalah tampilan awal *PHP*.

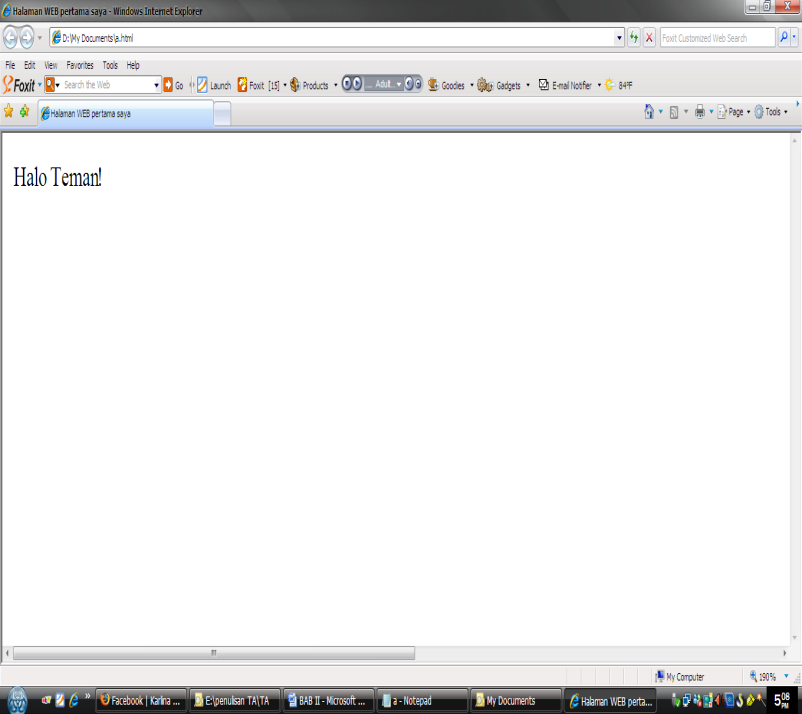


**Gambar 2.1 Halaman Awal *PHP* (** [**http://localhost**](http://localhost) **)**

**2.2 HTML**

*HTML* adalah bahasa markup yang umum digunakan. Kepopuleran *HTML* disebakan karena *HTML* ini mudah digunakan. Pembuatan dokument *Web* dengan  *HTML* dapat dilakukan dengan mudah dan cepat. Dokumen *Web* dapat tersaji dengan cepat ke banyak pembaca di seluruh dunia sekaligus*. HTML* mudah melakukan kontrol terhadap tampilan halaman *Web* baik berupa teks, gambar, suara, animasi maupun video. *HTML* merupakan singkatan dari *Hypertex Markup Language*. (Bheta, Sidik, Ir & Husni I, Ir, M. Eng, 2007). *HTML* digunakan untuk membangun halaman *Web*. Sekalipun banyak orang menyebutnya sebagai bahasa pemrograman, *HTML* sebenarnya sama sekali bukan bahasa pemrograman, hal ini terlihat dari namanya, *HTML* adalah suatu bahasa Markup. *HTML* digunakan untuk melakukan mark-up (penandaan) terhadap sebuah dokumen teks. Tanda tersebut digunakan untukmenentukan format atau style dari teks yang ditandai halaman *web* dibangun oleh  kode-kode *HTML* yang disebut juga dengan tag-tag *HTML* Tag yang harus di miliki oleh halaman *Web* adalah:

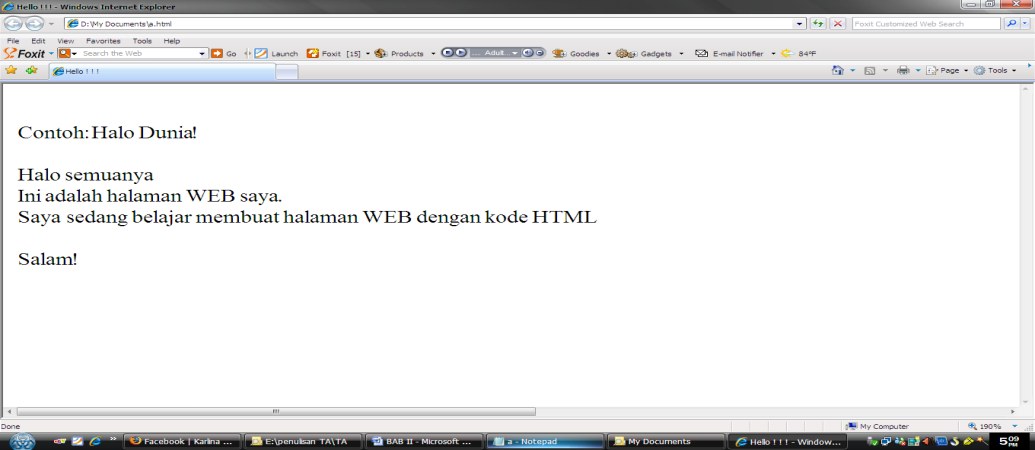
<HTML>  
<HEAD>  
<TITLE>Halaman WEB pertama saya</TITLE>  
</HEAD>  
<BODY>  
Halo Teman!  
</BODY>  
</HTML>  
Dari kode diatas menghasilkan output



**Gambar 2.2 Output Contoh Program *HTML***

Di antara kedua tag tadi adalah tempat memasukkan apa saja ke dalam halaman *web*. Pada bagian HEAD, kita memberikan data mengenai dokumen kita. Tag penutup selalu ada pada halaman tersebut. Paragraph/Baris Baru:

<P><BR>  
Contoh:  
<HTML>  
<HEAD>  
<TITLE>Hello ! ! !</TITLE>  
</HEAD>  
<BODY>  
Halo Dunia!<p>  
Halo semuanya<br> Ini adalah halaman WEB saya. <br>  
Saya sedang belajar membuat halaman WEB dengan kode HTML<p>  
Salam!  
</BODY>  
</HTML>



**Gambar 2.3 Output Contoh Program HTML II**

**2.3 MySQL**

*MySQL* adalah *Relational Database Management System* (*RDBMS*) yang didistribusikan secara gratis dibawah lisensi *General Public License* (GPL). Setiap orang bebas untuk menggunakan *MySQL,* namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. *MySQL* merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu *Structured Query Language (SQL). SQL* adalah sebuah konsep pengoperasian *database,* terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. (Prasetyo, Dwi, 2005). Keandalan suatu sistem *database* *(DBMS)* dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah *SQL,* yang dibuat oleh *user* maupun program - program aplikasinya. Sebagai *database server*, *MySQL* dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lebih cepat dari *PostgreSQL* dan lima kali lebih cepat dibandingkan *Interbase*. Selain itu *MySQL* juga memiliki beberapa keistimewaan, antara lain : (Prasetyo, Dwi, 2005)

1. Portability

*MySQL* dapat berjalan stabil pada berbagai *sistem operasi* seperti *Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga*, dan masih banyak lagi.

2. Open Source

*MySQL* didistribusikan secara *open source* (gratis), dibawah lisensi *GPL* sehingga dapat digunakan secara cuma - cuma.

3. Multiuser

*MySQL* dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4. Performance tuning

*MySQL* memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak *SQL* per satuan waktu.

5. Column types

*MySQL* memiliki tipe kolom yang sangat kompleks, seperti *signed* / *unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.

6. Command dan functions

*MySQL* memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam *query.*

7. Security

*MySQL* memiliki beberapa lapisan sekuritas seperti level *subnetmask*, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkripsi.

8. Scalability dan limits

*MySQL* mampu menangani *database* dalam skala besar, dengan jumlah *records* lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. Connectivity

*MySQL* dapat melakukan koneksi dengan client menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).

10. Localisation

*MySQL* dapat mendeteksi pesan kesalahan pada *client* dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk didalamnya.

11. Interface

*MySQL* memiliki *interface* (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).

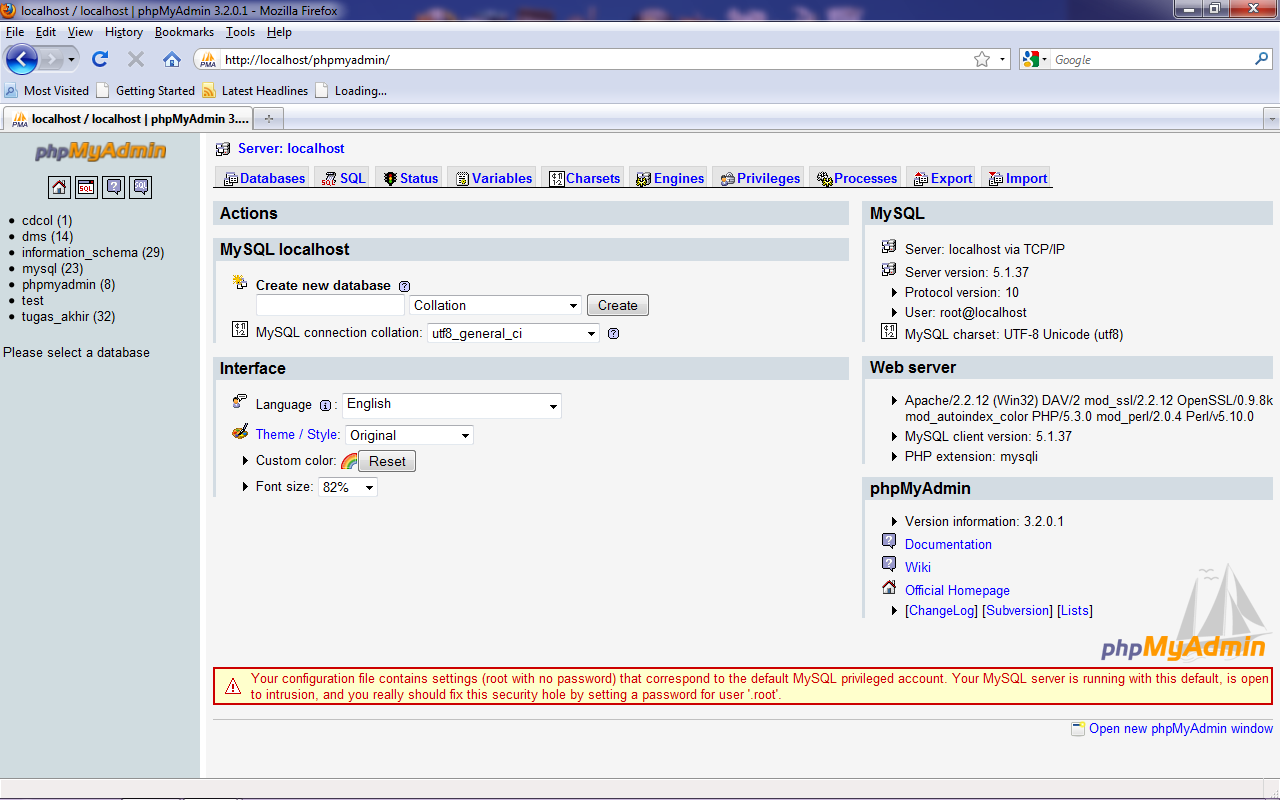
12. Clients dan tools

*MySQL* dilengkapi dengan berbagai *tool* yang dapat digunakan untuk *administrasi database*, dan pada setiap *tool* yang ada disertakan petunjuk *online*.

13. Struktur tabel

*MySQL* memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan *database* lainnya semacam *PostgreSQL* ataupun *Oracle*.

Berikut adalah tampilan *PHP*myadmin :



**Gambar 2.4 Tampilan *PHPMyadmin* (http://localhost)**

**2.4 OOP**

OOP atau kepanjangan dari *Object Oriented Programming* adalah sebuah cara pandang (paradigma) dalam memecahkan masalah di dunia nyata dianalogikan kedalam bentuk *object* dan struktur data yang didalamnya terdapat kumpulan *item* data disebut *variable* dan *method*/fungsi yang saling berinteraksi dalam sebuah struktur program. OOP juga merupakan paradigma pemrograman yang popular saat ini yang telah menggantikan teknik pemrograman berbasis prosedur. OOP ini dikembangkan pada permulaan tahun 1970. Pemrograman Berorientasi Objek (*Object Oriented Programming*/OOP) merupakan pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam *class - class* atau *object - object*. Setiap *object* dapat menerima pesan, memproses data, mengirim, menyimpan dan memanipulasi data. Beberapa object berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya. Masing - masing object harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan *Object* yang lain. Jadi, secara mudah OOP dapat disimpulkan sebagai suatu teknik atau metode untuk menangani kompleksitas pembuatan program. Jika mengikuti aturan-aturan dalam metode tersebut maka program yang dibuat akan lebih mudah dikembangkan. (One Creative)

**2.5 MVC**

*Model View Controller* (MVC) adalah sebuah pola yang dapat membangun proyek perangkat lunak lebih efektif karena dilakukan dengan memilah komponen antara *Model*, *View*, dan *Controller* pada bagian - bagian dalam proyek. Hampir perkembangan *mvc* dalam dunia pemrograman adalah  perkembangan dimana setiap orang bisa belajar dan membuat program yang *enterprise* (yang berbobot) tanpa harus mahir dan belajar bertahun - tahun atau pun membutuhkan *teamwork* yang banyak. konsep *MVC* tidak hanya di gunakan dalam satu jenis program, bahkan sekarang bisa dikatakan hampir menyeluruh di setiap jenis pemrograman, baik yang open source maupun yang berbayar. (Panada, 2012)

Dalam *MVC* dikenal tiga komponen pembangun.



**Gambar 2.5 Konsep MVC (Baba Studio, 2011)**

1. **Model**

Model adalah sebuah layer pada MVC yang merepresentasikan data yang digunakan oleh aplikasi sesuai dengan proses bisnis yang terjadi pada data tersebut, dengan memilahnya menjadi beberapa bagian terpisah kembali, seperti penampungan data, *persistence*, proses manipulasi, dan lain- lain. (Panada, 2012)

Kelebihan dengan adanya *Model* adalah :

1. Proses maintenance aplikasi

Penempatan detail data dan operasinya pada area yang ditentukan (Model) sehingga tidak tersebar pada   keseluruhan lingkup aplikasi.

1. Reusable Model

Penggunaan kembali aplikasi model oleh aplikasi lain yang disebabkan adanya pemisahan data dengan interface.

1. **View**

*View* adalah sebuah *layer* pada *MVC* yang mengandung keseluruhan detail dari implementasi *user interface* dengan melibatkan komponen grafis yang menyediakan representasi proses internal aplikasi dan meuntun alur interaksi user terhadap aplikasi. (Panada, 2012)

Kelebihan dari View adalah :

* Memudahkan penggabungan divisi desain dalam development team.
* Ketersediaan *multiple interface* dalam aplikasi.

1. **Controller**

Controller adalah sebuah layer pada MVC yang menyediakan detai alur program dan transisi layer, dan bertanggung jawab akan penampungan event yang dibuat oleh user dari view dan melakukan update terhadap komponen model menggunakan data dari user. (Panada, 2012)

Kelebihan dari Controller adalah :

* Komponen *view* dapat didesain tanpa harus memperhatikan bagian lain secara berlebih.
* Pemisahan jelas antara *presentation logic* dan *business logic*.

**2.6 UML**

*Unified Modelling Language (UML)* adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak (Sholiq, 2006). *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. Simbol – symbol yang terdapat pada *Unified Modelling Language* (*UML) :*

**Tabel 2.1 simbol – symbol UML.**

|  |  |  |
| --- | --- | --- |
| Simbol | Nama | Penjelasan |
|  | Actor | Sebuah peranan yang bias dimainkan oleh pengguna dalam interaksinya dengan system |
|  | Usecase | Abstraksi dari interaksi antara system dan actor |
|  | Uses | Untuk menghubungkan actor dan usecase atau actor dengan actor |
|  | Class | Himpunan objek – objek yang sejenis |
|  | Initial | Menandakan suatu titik awal |
|  | Final | Menandakan suatu titik akhir |
|  | Activity | Menyatakan suatu kegiatan yang terjadi |
|  | Decision | Pilihan untuk mengambil keputusan |
|  | Self massage | Proses yang dapat terjadi berulang-ulang |
|  | Time | Diagram yang mewakili waktu pada arah vertical, waktu dimulai dari atas ke bawah |
|  | Component | Menggambarkan fisik dari suatu system |
|  | Depence | Menggambarkan alur dari suatu component |

**UML mendefinisikan diagram-diagram berikut ini :**

* Usecase diagram
* Class diagram
* Activity diagram
* Sequence diagram
* Collaboration diagram
* Statechart diagram
* Component diagram
* Deployment diagram

***USE CASE DIAGRAM***

*Use case diagram* menyajikan interaksi antara *usecase* dan *actor.* Dimana, *actor* dapat berupa orang, peralatan, atau system lain yang berinteraksi dengan system yang sedang dibangun. *Usecase* menggambarkan fungsionalitas sistem atau persyaratan – persyaratan yang harus dipenuhi system dari pandangan pemakai.



**Gambar 2.6 *Usecase diagram*.**

Diagram *usecase* ini menunjukan interaksi antara *use case* dan *actor* untuk sistem ATM. Pada contoh ini, actor pelanggan menggunakan beberapa use case, antara lain: menarik uang, mendepositkan dana, mentransfer uang, mengecek saldo, membayar kredit, dan mengganti PIN. Petugas bank dapat mengganti PIN pelanggan. Use case membayar kredit memberikan arah panah ke *actor* sistem kredit, dimana merupakan *actor* berupa system lain yang menerima informasi/data dari sistem ATM.

***Class diagram***

*Class diagram* menunjukan interaksi antar kelas dalam sistem.. Dalam class diagram ini terdapat 4 *class*, diantaranya : pembaca kartu, layar ATM, *account* dan *dispenser* tunai. Berikut adalah penggambarannya :



**Gambar 2.7 *Class diagram* (Sholiq, 2006)**

Diagram diatas menunjukan hubungan antar kelas – kelas yang diimplementasikan oleh use case “menarik uang”.

**Activity diagram**

Activity diagram menggambarkan aliran fungsionalitas sistem. Pada tahap pemodelan bisnis, diagram aktivitas dapat digunakan untuk menunjukan aliran kerja bisnis. Dan dapat juga digunakan untuk menggambarkan aliran kejadian dalam *usecase*. Diagram aktivitas tidak perlu dibuat untuk setiap aliran kerja, tetapi diagram ini akan sangat berguna untuk aliran kerja yang komplek dan melebar.



**Gambar 2.8 *Activity diagram*.**

**Sequence diagram**

Sequence diagram digunakan untuk menunjukan aliran fungsionalitas dalam use case.



**Gambar 2.9 *Sequence diagram*** (Sholiq , 2006).

**Collaboration diagram**

Collaboration diagram menunjukkan informasi yang sama persis dengan *sequence* diagram, tetapi dalam bentuk dan tujuan yang berbeda. Pada *sequence* diagram, keseluruhan interaksi berdasarkan urutan waktu, tetapi pada *collaboration* diagram, interaksi antar *object* atau *actor* ditunjukkan dengan arah panah tanpa keterangan waktu. (Sholiq, 2006).



**Gambar 2.10 *Collaboration diagram*** (Sholiq, 2006).

***Statechart diagram***

*Statechart diagram* menyediakan sebuah cara untuk memodelkan bermacam – macam keadaan yang mungkin dialami oleh sebuah objek. (Sholiq, 2004). *Statechart diagram* digunakan untuk memodelkan tingkah laku dinamik sistem.



**Gambar 2.11 *Statechart diagram*** (Sholiq, 2006).

***Component diagram***

*Component diagram* menunjukan model secara fisik komponen perangkat lunak pada sistem dan hubungannya antar sistem. Masing – masing kelas dalam model akan dipetakan ke sebuah komponen kode pustaka. Setelah komponen dibuat, mereka ditambahkan dalam diagram komponen dengan memberikan relasi antara komponen – komponen. Relasi yang terjadi antar komponen hanya satu tipe relasi yaitu dependensi yang menunjukan ketergantungan *compile-time* dan *runtime* antara komponen – komponen tersebut.



**Gambar 2.12 *Component diagram*.**

***Deployment diagram***

*Deployment diagram* menampilkan rancangan fisik jaringan dimana berbagai komponen akan terdapat disana.



**Gambar 2.13 *Deployment diagram***