BAB II

DASAR TEORI

2.1 *Social Network*

Revolusi informasi telah melahirkan ekonomi baru yang terbentuk dari kumpulan aliran data, informasi, dan pengetahuan. Secara paralel, *Social Network* telah tumbuh kuat sebagai bentuk organisasi aktifitas manusia, istilah ini dikemukakan pertama kali tahun 1954 oleh Professor John Barnes. *Social Network* adalah suatu simpul dari individu, kelompok, organisasi, dan sistem terkait yang terikat dalam satu atau lebih hubungan saling ketergantungan. Termasuk juga berbagi dalam hal sesuatu yang berharga, visi, ide, kontak sosial, kekerabatan, konflik, bursa keuangan, perdagangan, keanggotaan bersama dalam organisasi , dan partisipasi kelompok dalam suatu kegiatan, berada di tengah - tengah berbagai macam aspek lain dari hubungan antar manusia [SER-09].

Berdasarkan penjelasan *social network* diatas perangkat lunak Jejaring Sosial Komunitas Peneliti dirancang dengan menggunakan konsep *social network* untuk menciptakan suatu interaksi di antara para Dosen. Interaksi antara Dosen satu dengan yang lain diharapkan memberikan keuntungan pada saat Dosen membutuhkan suatu informasi yang berkaitan dengan penelitian yang sedang dikerjakan.

2.2 Keuntungan membangun Aplikasi *Social Network* menggunakan *Object Oriented Programming*

Pemrograman Berorientasi Objek (*Object Oriented Programming*/OOP) adalah pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dijadikan dalam satu wadah yaitu dalam *class-class* atau *object-object*. Aplikasi dengan konsep *Social Network* adalah aplikasi dengan skala besar dan kompleks dengan kerumitan program sesuai dengan apa yang disediakan oleh aplikasi tersebut untuk menunjang interaksi yang terjadi antar *User* yang

menggunakan aplikasi *social network* tersebut. Sifat *Object Oriented Programming* yang fleksibel, program lebih mudah dipahami dan dibaca, dapat merubah, menambah atau mengurangi program sesuai kebutuhan tanpa mengganggu jalannya keseluruhan program sangat cocok sekali digunakan dalam kasus pembuatan aplikasi yang rumit dan kompleks seperti Aplikasi *Social Network* karena memberikan berbagai kemudahan kepada pemrogram seperti yang telah disebutkan diatas. [ZEG-96]

2.3 Penelitian Dosen pada Badan Penelitian dan Pengabdian Kepada Masyarakat

Proposal penelitian yang diajukan oleh Dosen Pengusul mengandung materi antara lain Judul Penelitian, Latar Belakang, Rumusan Masalah, Tujuan dan Manfaat, Metodologi Penelitian dan Daftar Pustaka. Anggota dan Ketua Tim adalah Dosen di lingkungan Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Data Dosen yang dibutuhkan dalam Penelitian yaitu Nama, NIP, Jurusan, Fakultas dan Bidang Keahlian. Dosen Pengusul dapat merekrut anggota Tim sesama Dosen pada Program Teknologi Informasi dan Ilmu Komputer walaupun berbeda bidang keahlian. Dosen Pengusul tidak selalu menjadi ketua tim dikarenakan telah menjadi Ketua Tim pada Penelitian yang lain. [BAD-13]

2.4 Gambaran Umum Perangkat Lunak Jejaring Sosial Komunitas Peneliti

Perangkat lunak yang akan dikembangkan dalam proyek skripsi ini adalah perangkat lunak Jejaring Sosial Komunitas Peneliti. Perangkat lunak Jejaring Sosial Komunitas Peneliti adalah perangkat lunak berbasis web dengan konsep jejaring sosial yang memungkinkan antar Dosen sebagai peneliti dapat mengetahui informasi mengenai topik – topik penelitian dari Dosen lain. Selain itu perangkat lunak Jejaring Sosial Komunitas Peneliti juga membantu para Dosen untuk lebih mudah dalam mencari rekan sesama Dosen untuk diajak ataupun bergabung dalam suatu penelitian. Perangkat lunak ini nantinya akan

diimplementasikan dahulu untuk para Dosen di Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya Malang.

Sistem perangkat lunak Jejaring Sosial Komunitas Peneliti sistem akan dirancang dan dibangun sesuai kebutuhan yang diperlukan yaitu bagaimana sesama Dosen sebagai peneliti dapat mengetahui informasi mengenai topik penelitian dari Dosen lain. Informasi penelitian yang ditampilkan kepada Dosen sebatas topik penelitian dan bidang keahlian. Apabila Dosen ingin mengetahui lebih banyak mengenai penelitian tersebut dapat mengirimkan Pesan Pribadi kepada Dosen yang bersangkutan. Apabila topik penelitian di *post* dengan status Terbuka, maka Dosen yang memposting otomatis sebagai Dosen Pengusul dapat mengajak Dosen lain untuk bergabung dengan tim atau Dosen lain dapat mengirimkan permintan untuk bergabung kepada Dosen Pengusul. Ketua Tim akan dipilih oleh anggota tim setelah proses mengajak dan diajak untuk bergabung selesai dilaksanakan. Keuntungan dari menjadi Ketua dan anggota tim adalah dapat mengetahui detail dari topik penelitian yaitu latar belakang, rumusan masalah, manfaat, daftar pustaka dan metodologi penelitian. Tentu saja anggota yang diajak bergabung ataupun ingin bergabung harus sesuai persyaratan yang dideskripsikan oleh ketua tim. Perangkat lunak Jejaring Sosial Komunitas Peneliti memiliki 2 bagian utama yaitu, subsistem aplikasi Dosen dan subsistem aplikasi administrator.

a) Aplikasi Dosen

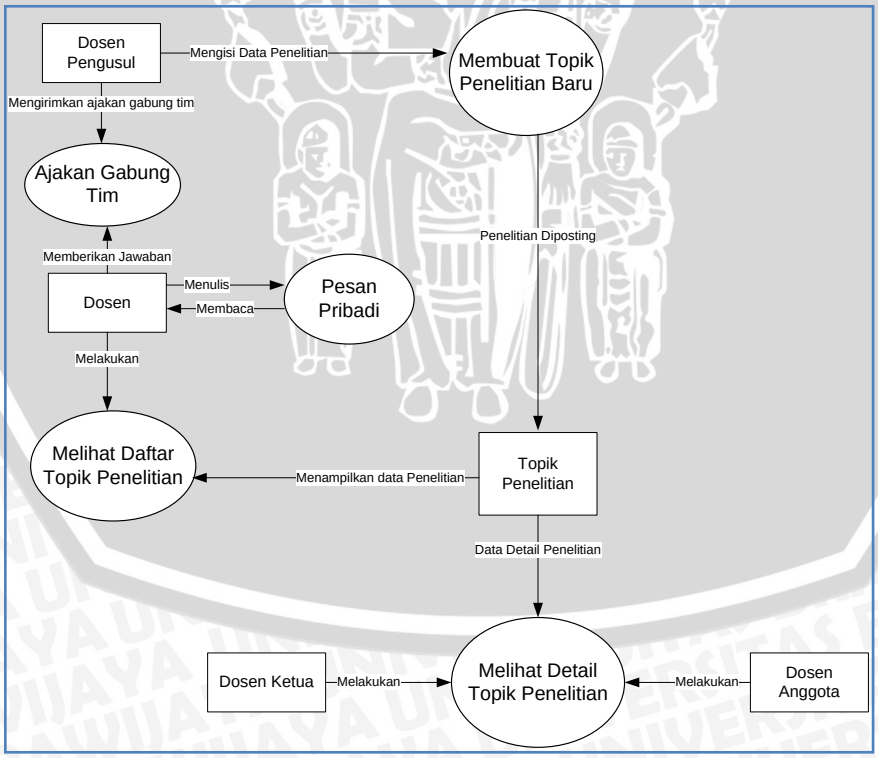
   Aplikasi Dosen dari perangkat lunak Jejaring Sosial Komunitas Peneliti adalah aplikasi yang menyediakan fitur agar para Dosen dapat melihat dan mengetahui topik penelitian yang diposting oleh Dosen lain. Ditambah dengan fitur tambahan untuk mencari rekan sesama Dosen untuk dijadikan Tim dan pesan pribadi untuk berkomunikasi dengan Dosen lain.

b) Aplikasi Administrator

   Aplikasi administrator dari perangkat lunak Jejaring Sosial Komunitas Peneliti adalah aplikasi yang digunakan untuk menonaktifkan dan mengaktifkan kembali Dosen, menghapus

topik penelitian yang sudah diposting, dan menghapus komentar yang sudah diposting oleh Dosen.

Gambaran proses bisnis perangkat lunak Jejaring Sosial Komunitas Peneliti diawali dengan proses membuat topik penelitian baru yang dilakukan oleh Dosen Pengusul dengan memasukkan data penelitian dan topik penelitian baru akan disimpan dan diposting pada proses melihat daftar topik penelitian. Dosen dapat melakukan proses melihat daftar topik penelitian dan untuk melihat detail dari topik penelitian maka harus menjadi Dosen Ketua atau Dosen anggota dengan cara bergabung dengan Tim. Dosen Pengusul dapat mengirimkan ajakan gabung tim kepada Dosen melalui proses Ajakan Gabung tim dan Dosen menjawab ajakan tersebut untuk menentukan apakah menjadi anggota tim atau tidak. Pesan Pribadi digunakan Dosen untuk berkomunikasi dengan Dosen lain. Untuk lebih jelasnya gambaran proses bisnis perangkat lunak Jejaring Sosial Komunitas Peneliti ditunjukkan pada Gambar 2.1 berikut ini.



Gambar 2.1 Gambaran Proses Bisnis Perangkat Lunak Jejaring Sosial Komunitas Peneliti

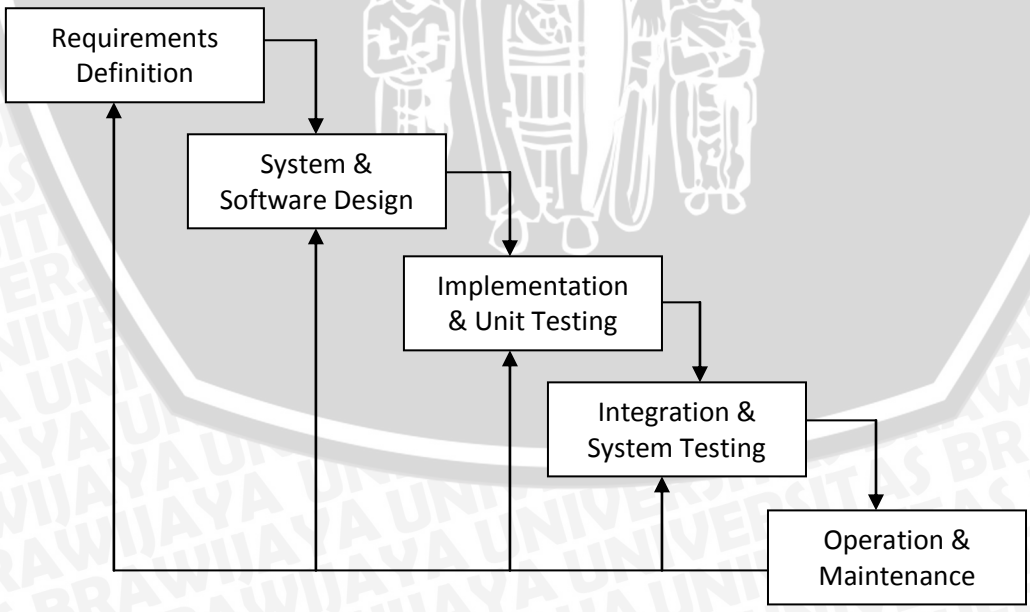
2.5 Model Proses Perangkat Lunak *Waterfall*

Seorang *software engineer* atau sekumpulan *software engineer* harus menggabungkan strategi pengembangan perangkat lunak yang meliputi proses, metode, dan alat bantu. Strategi ini yang kemudian sering disebut sebagai model proses (*process model*) atau paradigma rekayasa perangkat lunak (*software engineering paradigm*).

Model proses *waterfall* ini merekomendasikan pendekatan yang sistematis dan terurut (*systematic and sequential approach*) untuk pengembangan perangkat lunak yang dimulai dari analisis kebutuhan (*requirement analysis*), perancangan (*design*), implementasi (*coding*), pengujian (*testing*), dan pemeliharaan (*support*/*maintenance*) [PRE-10:28].

Kelebihan menggunakan model proses *waterfall* adalah dokumentasi dilakukan pada setiap fase sehingga berkesinambungan dengan model proses teknis lainnya. Kekurangan model proses ini adalah partisi proyek yang tidak fleksibel dan komitmen harus dibuat pada awal proses sehingga membuatnya sulit untuk menghadapi perubahan kebutuhan dari *customer* [SOM-10:67]:

Proses pengembangan perangkat lunak dalam skripsi ini menggunakan model proses *waterfall* seperti terlihat pada gambar 2.2 [SOM-10:66]:



Gambar 2.2 Model *Waterfall* menurut Ian Sommerville

Sumber : [SOM-07:66]

- *Requirements Definition*

Tahap pendefinisian kebutuhan meliputi pengumpulan keperluan sistem yang dibutuhkan secara lengkap [SOM-10:66]. Objektif dari analisis berorientasi objek (*Object Oriented Analysis*/OOA) adalah untuk mengembangkan sebuah model yang menjelaskan perangkat lunak bekerja sesuai kebutuhan yang didefinisikan oleh *customer* [PRE-10:572]. Diagram pemodelan aplikasi keseluruhan berdasarkan analisis kebutuhan yang dilakukan digambarkan dengan *use case diagram* [PRE-10:581].

- *System & Software Design*

Tahapan ini akan dikerjakan setelah pendefinisian kebutuhan pada tahap awal selesai [SOM-10:66]. Perancangan berorientasi objek (*Object Oriented Design*/OOD) mentransformasikan model yang dibuat menggunakan OOA. OOD dibagi dalam dua aktifitas besar yaitu, *system design* dan *object design*. *System design* membuat *product architecture*, mendefinisikan *layer-layer* yang melakukan fungsi sistem tertentu dan mengidentifikasi kelas-kelas yang dienkapsulasi oleh subsistem yang berada pada setiap *layer*. Sedangkan *object design* bertumpu pada detail internal individu – individu kelas, mendefinisikan atribut-atribut, operasi-operasi dan *message* [PRE-10:603].

- *Implementation & Unit Testing*

Setelah tahapan design selesai maka akan diterjemahkan ke dalam kode-kode pada bahasa pemrograman yang diinginkan [SOM-10:66]. Pengujian unit berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak, yakni modul. Pengujian unit biasanya berorientasi pada *white-box*, dan langkahnya dapat dilakukan secara paralel untuk model bertingkat Semua jalur independen (jalur dasar) yang melalui struktur kontrol dipakai sedikitnya satu kali [PRE-10:485].

White-box testing atau *glass-box testing* merupakan sebuah metode perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedural untuk memperoleh kasus uji [PRE-10:444]. Ada dua jenis pengujian

yang termasuk *white-box testing* yaitu *basis path testing* dan *control structure testing*. *Basis path testing* memungkinkan perancang kasus uji memperoleh ukuran kompleksitas logis dari sebuah perancangan prosedural dan menggunakan pengukuran ini sebagai pedoman untuk mendefinisikan *basis set* dari jalur eksekusi (*execution path*). *Test case* yang dilakukan untuk menggunakan *basis set* tersebut dijamin untuk menggunakan setiap *statement* di dalam program paling tidak sekali selama pengujian.

- *Integration & System Testing*

Setelah program selesai dibuat maka akan diujikan sebelum di implementasikan melalui tahapan *testing* (uji sistem) sehingga *user* (pengguna) dapat mengetahui kehandalan dari sistem yang dibuat [SOM-10:66]. Pengujian integrasi adalah teknik sistematis untuk mengkonstruksi struktur program sambil melakukan pengujian untuk mengungkap kesalahan sehubungan dengan *interfacing*. Sasarannya adalah untuk mengambil modul yang dikenai pengujian unit dan membangun struktur program yang telah ditentukan oleh desain.

Integration testing berorientasi *black box* dan mempunyai dua pola pengujian yaitu integrasi *top-down* (*top-down integration)* dan integrasi *bottom-up* (*bottom-up integration)* [PRE-10:488]. *Black-box testing* atau *behavioral testing* berfokus pada persyaratan fungsional perangkat lunak [PRE-10:459]. Pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut (1) fungsi-fungsi yang tidak benar atau hilang. (2) kesalahan *interface*, (3) kesalahan dalam struktur data atau akses *database* eksternal, (4) kesalahan kinerja, (5) inisialisasi dan kesalahan terminasi. Pengujian didesain untuk menjawab pertanyaan-pertanyaan:

- Bagaimana validitas fungsional diuji ?
- Kelas input apa yang akan membuat *test case* menjadi baik ?
- Apakah sistem sangat sensitif terhadap harga input tertentu ?
- Bagaimana batasan dari suatu data diisolasi ?
- Kecepatan dan volume data apa yang dapat ditolerir oleh sistem ?
- Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem ?

Validasi dapat ditentukan dengan berbagai cara, tetapi definisi yang sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi dengan cara yang dapat diharapkan secara bertanggung jawab oleh pelanggan [PRE-10:495].

- *Operation & Maintenance*

Langkah ini merupakan langkah terakhir dalam model proses *waterfall* yang dilakukan agar sistem dapat digunakan oleh *user* (pengguna) [SOM-10:66].

2.6 *Unified Modelling Language*

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa – bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C [DHA-06:2].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

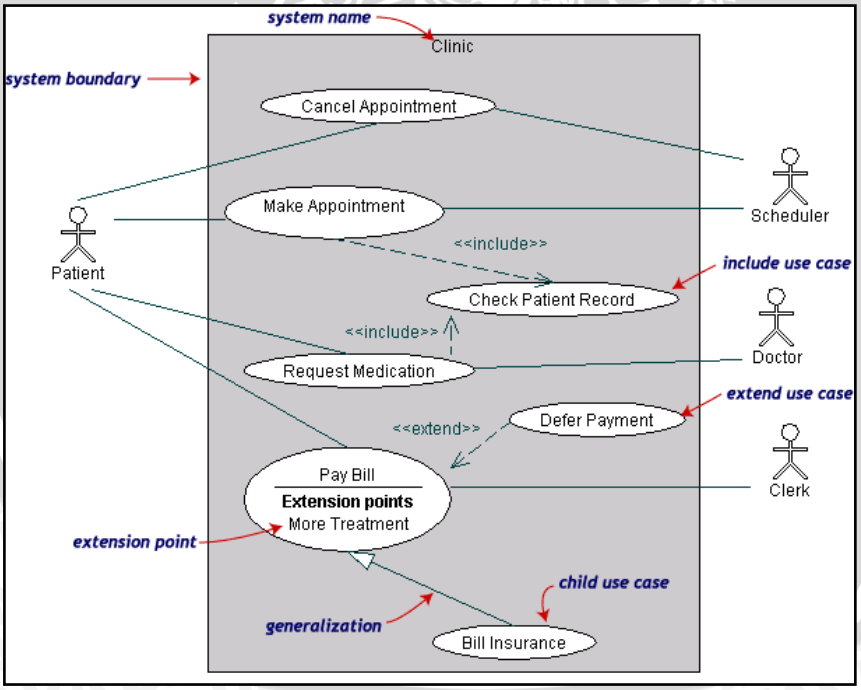
Dalam UML pemodelan digambarkan dengan penggunaan diagram – diagram yang masing – masing memiliki fungsi dan makna sendiri – sendiri. Diagram – diagram ini dapat digunakan sesuai dengan kebutuhan saat kita akan

merancang perangkat lunak. Beberapa diantaranya adalah diagram *use case*, diagram *class* dan diagram *sequence*.

2.7 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu [DHA-06:4].

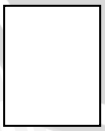
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

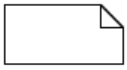


Gambar 2.3 Contoh *use case diagram*

Sumber : [DHA-06:5]

Tabel 2.1 Keterangan simbol - simbol *use case diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya sebagai elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / *ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.

9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

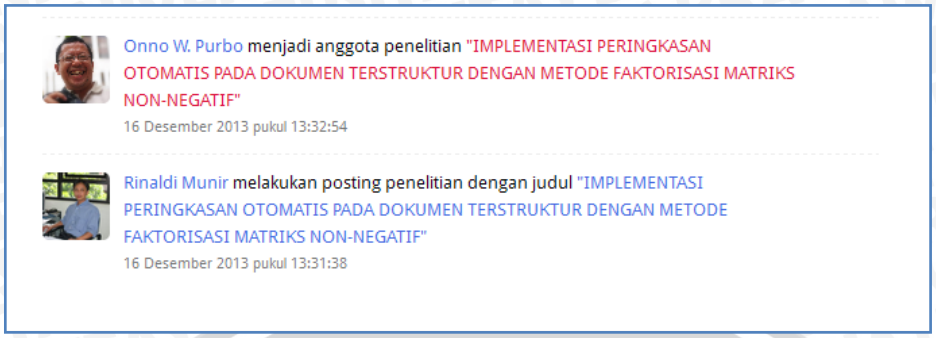
2.8 Penggunaan *Pseudo Selector* pada CSS untuk Implementasi Antarmuka

Untuk mengatur implementasi antarmuka agar sesuai dengan rancangan antarmuka digunakan *Cascading Style Sheets* (CSS) untuk mengatur format pada struktur konten yang dibangun dengan HTML. Format struktur yang diatur mulai dari warna, jenis Font, besar tulisan dan lain sebagainya. Di dalam CSS digunakan *Pseudo Selector* untuk mempermudah pengaturan struktur konten. Misalkan pada penggunaan link pada tulisan tertentu, apabila *cursor* mengarah ke tulisan tersebut maka warna tulisan akan berubah. Berikut contoh penggunaan *Pseudo Selector* pada CSS pada Gambar 2.4.

```
a:hover {
        color: crimson;
}
```

Gambar 2.4 *Pseudo Selector* pada CSS

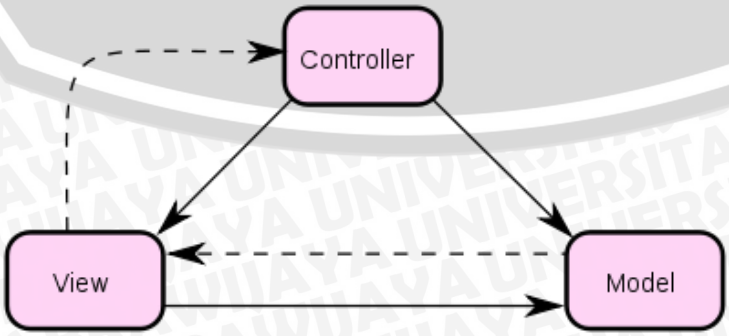
Pseudo Selector diatas mengatur sebuah tulisan yang diberi *link* akan berubah warna menjadi *crimson*(merah) apabila *cursor* mengarah pada tulisan tersebut. Akan ditunjukkan pada gambar 2.5 bagaimana Tulisan berubah dari biru menjadi *crimson*. [COL-06]



Gambar 2.5 Hasil dari *Pseudo Selector* pada CSS

2.9 *Model – View – Controller* (MVC) pada *Framework CodeIgniter*

Codeigniter adalah *framework* PHP yang dibuat berdasarkan kaidah *Model-View-Controller*(MVC). *Model-View-Controller* atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*). Dalam implementasinya kebanyakan *framework* dalam aplikasi *website* adalah berbasis arsitektur MVC. Bagian dari MVC adalah *Model* mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain. *View* adalah bagian yang mengatur tampilan ke pengguna. Bisa di katakan berupa halaman *web*. *Controller* merupakan bagian yang menjembatani *model* dan *view*. *Controller* berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman *web*. Gambar 2.6 menunjukkan MVC pada Framework CI. [Ell-12]



Gambar 2.6 MVC pada Framework CI