APLIKASI WEB PENELUSURAN DATA ALUMNI UNIVERSITAS BRAWIJAYA (TRACER STUDY)

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANGGITA YUAN R.

NIM. 0310630017

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2009

APLIKASI WEB PENELUSURAN DATA ALUMNI UNIVERSITAS BRAWIJAYA (TRACER STUDY)

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANGGITA YUAN R.

NIM. 0310630017

Telah Diperiksa dan Disetujui Oleh:

Arief Andy Soebroto, ST., M.Kom	**R. Arief Setyawan, ST., MT**
NIP. 132 231 567	**NIP. 132 231 713**

LEMBAR PENGESAHAN

APLIKASI WEB PENELUSURAN DATA ALUMNI
UNIVERSITAS BRAWIJAYA
(TRACER STUDY)

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANGGITA YUAN R.
NIM. 0310630017

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 14 Agustus 2009

Penguji 1	Penguji 2
Ir. Muhammad Aswin, MT	Suprapto, ST,. MT
NIP. 131 879 045	NIP. 132 149 320

Penguji 3

Adharul Muttaqin, ST., MT
NIP.

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir. Heru Nurwarsito, M.Kom
NIP. 131 879 033

PENGANTAR

Puji syukur penulis panjatkan ke hadapan Allah SWT karena berkat rahmat dan hidayahNya-lah penulis dapat menyelesaikan Tugas Akhir yang berjudul: "**Aplikasi *Web* Penelusuran Data Alumni Universitas Brawijaya (*Tracer Study*)".** Hanya kepada-Nya kita menyembah dan memohon. Teriring doa keselamatan untuk Rasulullah Muhammad SAW, keluarga, sahabat serta seluruh umatnya. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Konsentrasi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar besarnya kepada semua pihak yang telah memberikan bantuan baik material maupun non-material sehingga Tugas Akhir ini dapat terselesaikan dengan baik dan tepat pada waktunya. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada:

1. Bapak Ir. Heru Nurwarsito, M.Kom dan Bapak Rudy Yuwono, ST., M.Sc. selaku Ketua dan Sekretaris Jurusan Teknik Elektro serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya
2. Bapak Arief Andy Soebroto, ST., M.Kom. dan R. Arief Setyawan, ST., MT selaku pembimbing penulis dalam menyelesaikan tugas akhir ini.
3. Kedua Orang Tua penulis dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a demi terselesainya tugas akhir ini.
4. Seluruh Dosen Teknik Elektro Unibraw atas kesediaan membagi ilmunya kepada penulis.
5. Sahabat-sahabatku Hafid, Fitri, Memi, dan Ririk yang selalu memberikan semangat dan bantuan.
6. Teman-temanku Teknik Elektro Unibraw khususnya angkatan 2003, terima kasih atas segala bantuannya selama menjadi mahasiswa.
7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat. Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya

Malang, Agustus 2009

Penulis

ABSTRAK

ANGGITA YUAN R. 2009. : Aplikasi *Web* Penelusuran Data Alumni Universitas Brawijaya (*Tracer Study*). Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : Arief Andy Soebroto, ST., M.Kom dan R. Arief Setyawan, ST., MT.

Kemudahan dalam mendapatkan informasi semakin dibutuhkan dalam kegiatan-kegiatan manusia, baik oleh perorangan, perusahaan, ataupun instansi pendidikan dalam menjalankan kegiatannya. Perkembangan teknologi juga menuntut Universitas Brawijaya untuk mengembangkan UB dengan menyediakan sarana bagi alumninya untuk memberikan maupun mencari informasi tentang almamaternya. Jumlah alumni yang setiap tahunnya terus bertambah membutuhkan suatu sistem basis data sebagai media penyimpanan data dalam bentuk digital. Data-data tersebut harus dapat diakses, dimodifikasi, dan dianalisa dengan mudah. Untuk dapat mengakses data-data tersebut diperlukan suatu perangkat lunak yang mampu memilih dan mensortir data-data yang diperlukan dengan mudah. Karena data-data alumni sebelumnya belum begitu banyak dan belum tersimpan dengan baik, maka diperlukan studi penelusuran bagi alumni yang telah cukup lama lulus dari UB.

Perancangan dan pengimplementasian Aplikasi *Web* Penelusuran Data Alumni UB (*Tracer Study*) dilakukan dengan menggunakan bahasa pemrograman PHP (versi 5.2.5) dan basis data MySQL (versi 5.0.51a). *Web* Penelusuran Data Alumni UB terdiri dari empat kategori akses sistem antara lain, kategori Admin, kategori Alumni, kategori Fakultas dan Jurusan, dan kategori *user* eksternal.

Pengujian Aplikasi *Web* Penelusuran Data Alumni UB (*Tracer Study*) dilakukan pada setiap aplikasi sistem untuk mengetahui proses yang dilakukan oleh tiap-tiap aplikasi sistem tersebut. Hasil dari pengujian aplikasi sistem dapat diketahui bahwa aplikasi sistem pada *Web* Penelusuran Data Alumni UB (*Tracer Study*) dapat melakukan proses sesuai dengan kegunaannya masing-masing. Pengujian juga dilakukan terhadap koneksi basis data dan waktu akses *Query* yang digunakan oleh *Web* Penelusuran Data Alumni UB (*Tracer Study*). *Web* Penelusuran Data Alumni UB (*Tracer Study*) yang dihubungkan dengan koneksi basis data MySQL pada *port* 3306 dapat melakukan proses manipulasi terhadap data-data di dalam basis data. Pengujian waktu akses *Query* dilakukan pada masing-masing tabel dengan 500, 1000, 2000, dan 4000 data entri dan 10 kali percobaan. Dari hasil pengujian yang dilakukan terhadap koneksi basis data dan waktu akss *Query* menunjukkan bahwa *Web* Penelusuran Data Alumni UB (*Tracer Study*) dapat berfungsi dengan baik.

Kata Kunci : Penelusuran Data, Alumni, HTML, PHP, MySQL, *Web server* Apache.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sejak berdiri tahun 1963 hingga saat ini, Universitas Brawijaya (UB) telah meluluskan ribuan alumni yang tersebar di seluruh penjuru tanah air bahkan di luar negeri. Saat ini jumlah mahasiswa UB lebih dari 27.000 mahasiswa dengan penerimaan mahasiswa setiap tahun sekitar 7.000 mahasiswa masing-masing fakultas (dari 10 fakultas yang ada). Dengan demikian diperkirakan UB akan meluluskan sekitar 5.600 orang setiap tahunnya [ANO-08]. Sampai saat ini cara menyimpanan data alumni di UB belum tertata dengan baik. Pada periode-periode awal berdirinya UB, data tentang alumni bahkan tidak begitu diperhatikan. Keadaan yang seperti ini tidak bisa dibiarkan terus menerus karena akan menyebabkan kesulitan bagi UB untuk mendata alumninya yang terus bertambah.

Bagi alumni yang baru lulus seringkali kesulitan untuk mendapatkan informasi dari para alumni terdahulunya. Informasi yang dibutuhkan antara lain perusahaan-perusahaan yang membuka lowongan, gaji yang diterima, ataupun informasi lainnya.

Berdasarkan uraian di atas maka perlu dipikirkan untuk membuat suatu basis data yang berisi data-data tentang alumni UB. Data-data tersebut harus dapat diakses, dimodifikasi, dan dianalisa dengan mudah. Proses tersebut dapat dipermudah dengan menyediakan penyimpanan data dalam bentuk digital. Untuk dapat mengakses data-data tersebut diperlukan suatu perangkat lunak yang mampu memilih dan mensortir data-data yang diperlukan dengan mudah. Karena data-data alumni sebelumnya belum begitu banyak dan belum tersimpan dengan baik, maka diperlukan studi penelusuran bagi alumni yang telah cukup lama lulus dari UB. Sesuai dengan perkembangan teknologi informasi, maka selain pengumpulan data dilakukan dengan menyebarkan formulir isian, dapat pula dilakukan dengan memasang formulir isian tersebut pada suatu *website* di internet. Pemasangan formulir pada suatu *website* dapat mempermudah proses pengolahan

formulir data yang telah diisi karena alumni dapat secara langsung mengisikan data mereka dengan membuka *website* yang dimaksud.

1.2 Rumusan Masalah

Dengan memperhatikan permasalahan–permasalahan tersebut maka rumusan masalah ditentukan sebagai berikut:

1. Merancang *web tracer study* dengan menggunakan bahasa pemrograman *server-side* PHP 5.2.5 yang berbasis objek dan Basis data MySQL 5.0.51a.
2. Merancang aplikasi *web* untuk perbaikan kurikulum dan melihat profil alumni UB.
3. Merancang aplikasi untuk mengetahui kebutuhan pasar dan perkembangan teknologi terhadap lulusan UB.
4. Merancang sebuah aplikasi untuk mempermudah mengakses, meng-*update*, dan mengolah data alumni dengan menggunakan perangkat lunak yang mudah dioperasikan.
5. Mengimplementasikan dan menguji sistem yang telah dirancang.

1.3 Batasan Masalah

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam penyusunan tugas akhir ini antara lain:

1. Pembahasan difokuskan pada perancangan dan pembuatan aplikasi *web* dan berkomunikasi dengan sistem *database* MySQL.
2. Data yang digunakan mengacu pada data alumni yang sudah ada dalam *database* alumni UB.
3. Sistem informasi dibuat dengan menggunakan Sistem Operasi Microsoft Windows XP Professional Service Pack 2, XAMPP for Windows Version 1.6.6a yang mencakup program aplikasi, Web Server Apache 2.2.4 (Win32), database MySQL 5.0.51a dan bahasa pemrograman PHP 5.2.5.

1.4 Tujuan

Tujuan akhir dari skripsi ini adalah bagaimana merancang suatu aplikasi *web* menggunakan PHP 5 sebagai bahasa pemrograman yang berbasis OOP (*Object Oriented Programming*) untuk membantu mengumpulkan data-data alumni UB dan membuat *database* alumni sehingga akan mempermudah dalam pengelolaan dan pengembangan UB di masa yang akan datang.

1.5 Manfaat

Manfaat yang bisa didapatkan dari tugas akhir ini adalah:

1. Mempunyai kontribusi dalam kemudahan dalam mencari data alumni apabila diperlukan.
2. Mempermudah alumni yang baru lulus untuk menghubungi alumni yang sudah bekerja pada berbagai bidang tentang informasi lapangan kerja yang ada.
3. Apabila akan mengadakan kegiatan yang melibatkan alumni maka akan dapat dengan mudah menghubungi alumni yang bersangkutan.
4. Untuk pengembangan UB di masa yang akan datang, sehingga dari data alumni dapat dianalisa tentang relevansi antara pendidikan yang diperoleh selama kuliah dengan dunia kerja.

1.6 Sistematika Pembahasan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Menjelaskan tentang kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi ini.

BAB III Metodologi

Berisi tentang metode penelitian yang digunakan dalam perancangan dan pengujian sistem.

BAB IV Analisis kebutuhan dan Perancangan

Membahas tentang analisa kebutuhan dari sistem dan kemudian merancang hal-hal yang berhubungan dengan analisa tersebut.

BAB V Implementasi

Mengimplementasikan hasil perancangan ke dalam algoritma-algoritma dan kode-kode program dalam bahasa pemrograman.

BAB VI Pengujian

Memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VII Kesimpulan dan Saran

Memuat kesimpulan dan saran-saran.

BAB II

TINJAUAN PUSTAKA

Pada bab ini dijelaskan tentang kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi ini. Kajian pustaka diperlukan untuk melakukan kajian teradap karya ilmiah yang berkaitan dengan skripsi ini. Kajian pustaka yang dipaparkan adalah penelusuran data alumni berbasis *web*.

Dasar teori yang diperlukan berdasar kajian pustaka untuk penyusunan skripsi ini adalah : basis data, teori perancangan perangkat lunak, teori dasar *web*, dan jaringan komputer.

2.1 Penelusuran Alumni

Penelusuran alumni adalah sebuah proses yang terus-menerus karena alumni dari suatu institusi pendidikan selalu bertambah setiap tahun selama institusi tersebut masih eksis. Penelusuran alumni sangat berguna bagi institusi pendidikan sebagai bahan koreksi diri untuk perbaikan institusi di masa yang akan datang. Metode pengumpulan data dalam penelusuran alumni ada bermacam-macam. Pengumpulan data dapat dilakukan melalui wawancara, pengambilan dari arsip-arsip yang ada, penyebaran formulir-formulir yang harus diisi oleh alumni. Penyebaran formulir dapat dilakukan dengan mengirimkan melalui pos, dititipkan ke tempat kerja alumni, disebarkan melalui alumni yang memiliki akses luas ke alumni lainnya, dan melalui jaringan internet.

Data yang telah terkumpul selanjutnya dianalisa. Analisis dilakukan sesuai dengan keperluan, biasanya analisis dilakukan terhadap lama studi, indeks prestasi, lama tunggu untuk memperoleh pekerjaan, besarnya gaji yang diterima pada waktu pertama kali bekerja, kesesuaian bidang pekerjaan dengan latar belakang pendidikan yang diterima, dan lain-lain. Hasil analisis disajikan dalam bentuk tabel-tabel, grafik dan sebagainya [SUH-04].

Dari kondisi yang ada maka diambil topik penelusuran data alumni berbasis *web* untuk memberikan kemudahan dalam melakukan penelusuran data dan informasi alumni.

Sebagai penunjang kajian pustaka "Penelusuran Alumni" untuk penyelesaian skripsi ini, maka diperlukan dasar teori yang terdiri dari : basis data, teori perancangan perangkat lunak, teori dasar *web* dan jaringan komputer.

2.2 Basis Data

Pada skripsi ini akan menggunakan basis data sebagai dasar teori. Basis data diperlukan sebagai penyimpan kumpulan data elektronik alumni UB. Hal tersebut akan memberikan kemudahan dalam pengelompokkan dan pengaksesan data.

2.2.1 Definisi Basis Data

Basis data didefinisikan sebagai kumpulan data dalam media magnetik yang terstruktur dan diorganisasikan dengan cara tertentu sesuai dengan kebutuhan akan informasi. Sedangkan sistem basis data adalah sebuah sistem yang komponennya terdiri dari basis data dan kumpulan program untuk mengaksesnya. Sistem basis data berfungsi untuk mengatasi masalah-masalah yang timbul dalam pengelolaan data. Masalah-masalah tersebut antara lain [SUH-04]:

- **Redundansi dan inkonsistensi data**

  Jika *file-file* data dan program aplikasi dibuat oleh lebih dari satu *programmer*, *file-file* tersebut mungkin akan mempunyai format yang berbeda dan program aplikasi yang ditulis mungkin menggunakan beberapa bahasa pemrograman yang berbeda pula. Selain itu, beberapa informasi mungkin terduplikasi dibeberapa *file* sehingga terjadi redundansi data. Kasus redundansi menyebabkan kebutuhan akan *storage* menjadi besar. Selain itu, redundansi juga menyebabkan mudahnya timbul inkonsistensi data bila pemutahiran data yang terduplikasi tidak dilakukan secara menyeluruh.

- **Kesulitan pengaksesan data**

  Kadangkala lingkungan pengolahan *file* tidak mendukung untuk mengambil data secara efektif dan efisien. Sebagai contoh, andaikan seorang karyawan bank perlu data seluruh kastamer yang tinggal di kota tertentu, dan kebutuhan akan hal tersebut tidak diantisipasi pada saat perancangan sistem basis data, maka dia akan mempunyai dua pilihan yaitu mengambil data secara manual atau membuat program aplikasi khusus untuk melakukan hal itu. Kedua alternatif tadi sama-sama tidak memuaskan. Apabila sebagai solusinya dibuatkan program aplikasi, dan bila beberapa hari kemudian si karyawan tersebut menambahkan kriteria baru untuk menampilkan data kastamer yang dimaksud, misalnya hanya kastamer dengan saldo seratus juta, maka karyawan tadi kembali dihadapkan pada dua alternatif yang sama-sama tidak memuaskan.

- **Isolasi data**

  Isolasi data terjadi jika data-data tersebar di beberapa *file* dan *file-file* tersebut memiliki format yang berbeda. Masalah ini menimbulkan kesulitan dalam pembuatan program aplikasi baru, untuk mengambil data yang tepat.

- **Pengaturan akses oleh banyak pemakai**

  Untuk meningkatkan performansi sistem dan untuk mendapatkan waktu tanggap yang cepat, beberapa sistem mengijinkan beberapa *user* untuk memutahirkan data secara simultan. Pada lingkungan yang seperti ini, proses pemutahiran yang dilakukan secara bersama-sama, bisa menyebabkan data menjadi tidak konsisten. Sebagai contoh, misalkan ada *account* x sebesar 50 rupiah dan pada waktu yang sama terdapat dua kastamer menarik dana dari *account* x tersebut masing-masing sebesar 10 dan 20 rupiah, maka hasil dari proses konkuren tersebut mungkin akan tidak benar, *account* yang baru bisa menjadi 40 rupiah atau 30 rupiah padahal seharusnya adalah 20 rupiah. Untuk mencegah kemungkinan yang seperti ini maka harus terdapat koordinasi antar program aplikasi.

- **Pengamanan data**

  Untuk pengamanan data, *user* sistem basis data tidak semuanya harus bisa mengakses seluruh data. Sebagai contoh, di suatu bank, petugas *payroll* hanya perlu tahu data tentang karyawan bank, dia tidak perlu mengetahui informasi tentang rekening kastamer.

- **Integritas data**

  Nilai data yang terdapat dalam basis data harus memenuhi batasan data tertentu. Sebagai contoh, saldo minimal setiap kastamer bank tidak boleh kurang dari batas yang sudah ditentukan bank. Batasan tersebut dilakukan di dalam sistem dengan cara menambahkan kode yang tepat pada beberapa program aplikasi. Akan tetapi, ketika ada batasan baru yang perlu ditambahkan kedalam sistem, akan sulit untuk melakukannya apalagi jika batasan tersebut melibatkan beberapa item data dari beberapa *file* yang berbeda.

2.2.2 Perancangan Basis Data

Sebuah sistem basis data dibuat untuk mengatasi permasalahan-permasalahan yang nantinya akan memberikan kemudahan dan kecepatan dalam pengaksesan suatu informasi. Perancang sistem basis data harus memperhatikan struktur penyimpanan informasi dan mekanisme untuk memanipulasi informasi tersebut. Selain itu, perancang sistem basis data juga harus bisa menjamin keamanan informasi yang disimpan ketika terjadi *crash* atau terjadi manipulasi oleh orang yang tidak berhak. Jika data yang disimpan akan di-*share* kepada beberapa *users* maka sistem basis data juga harus menghindari keganjilan-keganjilan yang mungkin terjadi.

Perancang sistem basis data sebaiknya menyembunyikan kompleksitas data agar interaksi antara *user* dan sistem dapat dibuat sesederhana mungkin. Basis data yang ditampilkan kepada *user* merupakan gambaran abstrak dari data yang sesungguhnya.

Berdasarkan *level* abstraksi saat memandang suatu basis data, *user* sistem basis data dikelompokkan menjadi tiga *level* abstraksi [NUG-05]:

a. *Level* Fisik

Merupakan *level* abstraksi paling rendah. *Level* ini menggambarkan bagaimana data sebenarnya disimpan dan diorganisasikan secara fisik.

b. *Level* Konseptual

Pada *level* ini digambarkan keseluruhan data apa yang disimpan dalam basis data dan keterkaitan yang ada antar data. *Level* ini digunakan oleh *"database administrator"*.

c. *Level* Pandangan Pemakai

Merupakan *level* abstraksi tertinggi. *Level* ini menggambarkan hanya satu atau sebagian dari keseluruhan basis data, sesuai dengan kebutuhan pemakai.

Untuk memperjelas perbedaan antara ketiga level abstraksi data, kita analogikan dengan tipe data yang dikenal pada bahasa pemrograman. Hampir sebagian besar bahasa pemrograman mengenali tipe data *record*. Pada bahasa Pascal, deklarasi tipe data *record* ditulis sebagai berikut:

Type ***kastamer*** = *record*
Nama : *string*;
Alamat : *string*;
End;

Deklarasi di atas mendefinisikan sebuah tipe data *record* yang terdiri dari dua *field*. Setiap *field* mempunyai nama dan tipe.

Basis data di suatu bank mungkin mempunyai beberapa *record* yang lain, misalkan *record* ***rekening***, dengan *field-field* ***nomor*** dan ***saldo*** dan *record* ***karyawan***, dengan *field-field* ***nama*** dan **gaji**.

Pada level fisik, *record* ***kastamer***, ***rekening***, atau ***karyawan*** dijelaskan sebagai suatu blok lokasi penyimpanan yang berurutan. Pada *level* konseptual, setiap *record* dijelaskan seperti tersebut di atas, dan juga dijelaskan hubungan yang terdapat antara *record-record* tersebut. Pada *level* pandangan pemakai, beberapa "*view*" dari basis data dapat didefinisikan. Misalkan petugas kasir di

bank hanya dapat melihat informasi tentang rekening kastamer, mereka tidak dapat mengakses informasi tentang gaji karyawan [SUH-04].

Selain level abstraksi, dalam perancangan sistem basis data juga terdapat *tools* untuk menggambarkan data, hubungan data, arti data, dan batasan data. *Tools* tersebut adalah konsep model data. Terdapat beberapa jenis model data yang dapat dipergunakan, yaitu: model *logic* berbasis objek, model *logic* berbasis *record*, dan model data fisik.

Model *logic* berbasis objek digunakan untuk menjelaskan data pada *level* konseptual dan *level* pandangan pemakai. Jenis model data ini mempunyai karakteristik: mempunyai struktur yang fleksibel dan membolehkan batasan data dinyatakan secara eksplisit. Model data yang termasuk kedalam jenis ini adalah model *entity-relationship* dan model *object-oriented*.

Model *logic* berbasis *record* digunakan untuk mejelaskan *level* konseptual dan pandangan pemakai. Berbeda dengan model *logic* berbasis objek, model ini dapat menjelaskan struktur *logic* dan implementasi dari suatu basis data. Dalam model berbasis *record*, basis data disusun dalam beberapa *record* dengan format tetap. Setiap *record* mendefinisikan sejumlah *field*. Model data yang termasuk kelompok ini adalah model *relational*, model *network*, dan model hierarki [HAR-06].

Sebagai penutup tulisan ini marilah kita lihat beberapa model data yaitu model *entity-relationship* dan model *relational*.

2.2.2.1 Model *entity-relationship*

Model *entity-relationship* (model E-R) berisi kumpulan objek dasar, yang disebut *entity*, dan hubungan yang terdapat antar objek tersebut. *Entity* adalah objek yang dapat dibedakan dari objek yang lainnya berdasarkan atribut yang dimilikinya. Sebagai contoh, atribut ***nomor*** dan ***saldo*** menjelaskan *entity* ***rekening***. *Relationship* adalah hubungan yang terdapat antara beberapa *entity*. Contohnya *relationship* ***RekKas*** menghubungkan *entity* ***kastamer*** dan ***rekening***.

Kumpulan *entity* dan *relationship* yang mempunyai karakteristik sama, berturut-turut disebut *entity set* dan *relationship set* [SUH-04].

Selain *entity* dan *relationship*, model E-R dapat merepresentasikan batasan tertentu yang harus dipenuhi oleh nilai dalam basis data. Salah satu batasan yang penting adalah *mapping constrain* yang menetapkan jumlah *entity* yang dapat dihubungkan kedalam satu *relationship*.

Untuk merepresentasikan struktur basis data model E-R digunakan diagram E-R. Pada diagram E-R terdapat komponen-komponen [NUG-08]:

- Empat persegi panjang, merepresentasikan *entity*;
- Elips, merepresentasikan atribut;
- *Diamond*, merepresentasikan *relationship* ; dan
- Garis, menghubungkan atribut dengan *entity set* atau *entity set* dengan *relationship*.

2.2.2.2 Model Relational

Model *relational* merepresentasikan data dan hubungan antar data-data tersebut menggunakan sejumlah tabel. Setiap tabel mempunyai sejumlah kolom dengan nama yang unik. Tabel-tabel berikut ini memperlihatkan contoh basis data *relational* yang berisi data tentang kastamer dan rekening [SUH-04].

Tabel 2.1. Contoh Basis Data *Relational*

Nama	Alamat	Nomor
Lucyana	Jl. Anggrek 39	900
Shiver	Jl. Bogor 285	556
Kurniawan	Jl. Boko 45	801
Alwafi	Jl. Merak 41	647

Nomor	Saldo
900	55
556	100000
801	10533
647	105336

Sumber: [SUH-04]

2.2.2.3 SQL (*Structured Query Language*)

Stuctured Query Language (SQl) merupakan standar yang digunakan untuk mengakses database relational. Beberapa perangkat lunak yang menggunakan SQL sebagai perintah untuk mengakses data, di antanya DB2, Ingres, Informix, ORACLE, Microsoft Access, PostgreSQL, Rdb, Sybase dan lain-lain. SQL adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL dibuat sebagai bahasa yang dapat merelasikan antar database. Bahasa SQL ditulis langsung dalam sebuah program database sehingga seorang pengguna dapat lemihat langsung permintaan yang diinginkan, sekaligus melihat hasilnya.

SQL (*Stuctured Query Language*) dibagi menjadi dua bentuk *query*, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML) [NUG-08].

- **DDL (*Data Definition Language*)**

DDL adalah sebuah metode *query* SQL yang berguna untuk mendefinisikan data pada sebuah database, adapun *query* yang dimiliki adalah :

CREATE : digunakan untuk melakukan pembuatan tabel dan database

DROP : digunakan untuk melakukan penghapusan tabel maupun database.

ALTER : digunakan untuk melakukan pengubahan struktur tabel yang telah dibuat, baik menambah field (Add), mengganti nama field (change) ataupun menamakannya kembali (rename), serta menghapus (drop).

- **DML (*Data Manipulation Language*)**

DML adalah sebuah metode *query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *query* ini adalah untuk melakukan pemanipulasian database yang telah ada atau telah dibuat sebelumnya. Adapun *query* yang termasuk I dalamnya adalah ;

INSERT : digunakan untuk melakukan penginputan/pemasukan data pada tabel database

UPDATE : digunakan untuk melakukanpengubahan atau peremajaan terhadap data yang ada pada tabel

DELETE : digunakan untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa recordset.

2.2.2.4 MySQL (*My Structured Query Language*)

MySQL (*My Structured Query Language*) adalah sebuah program pembuat dan pengelola database atau yang sering disebut dengan DBMS (*DataBase Management System*) yang bersifat *open source*. MySQL sebenarnya produk yang berjalan pada *platform* Linux, dengan adanya perkembangan dan banyaknya pengguna serta lisensi dari database ini adalah *open source*, maka para pengembang kemudian merilis versi Windows.

Selain itu MySQL juga merupakan program pengakses database yang bersifat jaringan, sehingga dapat digunakan untuk aplikasi Multi User (banyak pengguna). Kelebihan lain dari MySQL adalah menggunakan bahasa *query* (permintaan) standar SQL (*Structured Query Language*). SQL adalah suatu bahasa permintaan yang terstruktur, SQL telah distandarkan untuk semua program pengakses database seperti Oracle, PosgreSQL, SQL Server, dan lain-lain.

Sebagai sebuah program penghasil database, MySQL tidak mungkin berjalan sendiri tanpa adanya sebuah aplikasi pengguna (*interface*) yang berguna sebagai program aplikasi pengakses database yang dihasilkan. MySQL dapat didukung oleh hampir semua program aplikasi baik yang *open source* seperti PHP maupun yang tidak *open source* yang ada pada platform Windows seperti Visual Basic, Delphi, dan lainnya [NUG-08].

2.3 Teori Perancangan Perangkat Lunak

Sebagai penunjang penyusunan skripsi ini dipelukan dasar teori perancangan perangkat lunak yang terdiri dari *Object Oriented Development* dan *Entity-Relationship Diagram*.

2.3.1 *Object Oriented Development*

Adalah suatu cara pengembangan perangkat lunak dan system informasi berdasarkan *abstraksi* objek-objek yang berada di dunia nyata. *Abstraksi* adalah

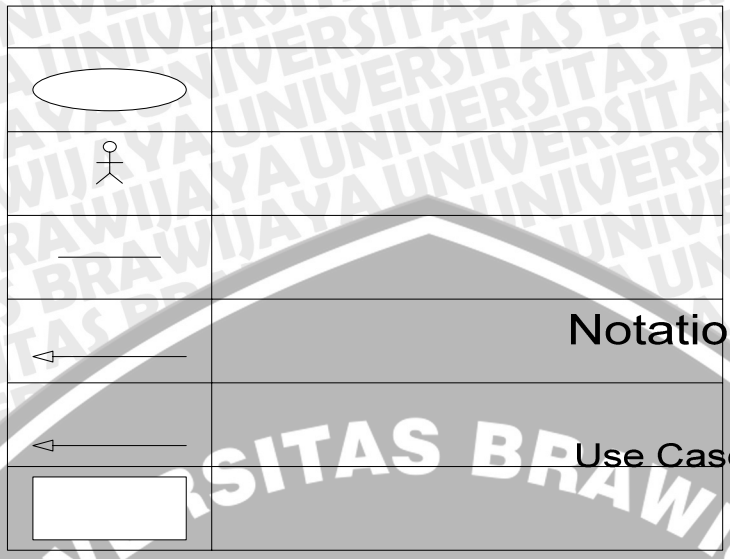
menemukan serta memodelkan fakta-fakta dari suatu objek yang penting bagi suatu aplikasi. Dengan metode pemrograman yang disebut dengan *Object Oriented Programming*, *Object Oriented Programming* adalah suatu cara baru untuk berfikir dan berlogika dalam menghadapi masalah-masalah yang-akan dicoba atasi dengan bantuan komputer [PRA-04].

Object Oriented programming memiliki *standard* bahasa pemodelan yang disebut dengan UML (*Unified Modelling Language*). UML menyediakan bermacam teknik dengan notasi grafik (diagram) memungkinkan *developers* untuk menyajikan *system model software* dalam bentuk yang berbeda dan dari sudut pandang yang berbeda. Meskipun UML terdiri dari 12 teknik, tetapi tidak seluruhnya diaplikasikan didalam *project* sistem, penggunaannya hanya sebagian saja terkait dengan kebutuhan sistem. Berikut akan dijelaskan beberapa teknik, terkait dengan tugas akhir ini [RAH-09].

2.3.1.1 *Use-case Diagram*

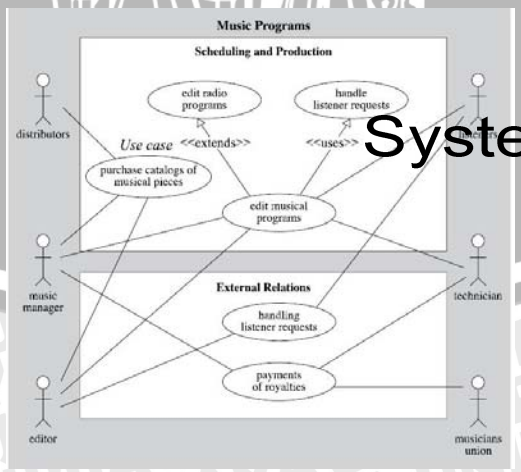
Diagram *Use-case* menunjukan interaksi antara sistem satu dengan sistem lainnya maupun dengan *user* sistem, yang disebut dengan aktor. Diagram ini menggambarkan siapa yang akan menggunakan sistem dan bagaimana *user* berinteraksi dengan sistem untuk mencapai tujuan yang diinginkan. Biasanya, diagram disertai dengan deskripsi narasi yang dijelaskan secara detil dengan cara yang terstruktur untuk langkah-langkah setiap interaksi. Teknik ini memungkinkan menciptakan/membuat deskripsi inisial kebutuhan *user*, sehingga nantinya perilaku sistem bisa ditetapkan dengan menggunakan arti yang lain. Contohnya, *sequence* atau *collaboration diagram*. Tabel 2.2 Menunjukan komponen-komponen di dalam *use-case* [SUS-07].

Tabel 2.2 Komponen *Use-case Diagram*



Sumber: [SUS-07]

Gambar 2.1 memperlihatkan diagram *use case* untuk contoh program *music*. Diagram tersebut terdiri dari 6 *use case*, yang dibagi dalam 2 subsistem, yang masing-masing subsistemnya berisi : **Assignment and Productions** dan **External Relations.** Pembagian ini memungkinkan 2 *team* mengembangkan sistem. Seperti yang bisa di lihat subsistem **"Assignment and Productions"** terdiri atas 4 *use case*, termasuk **"Edit Musical Program"** *use case* ini melanjutkan *use case* **"Edit Radio Program"** dan menggunakan "*Handle Listener Request*" Subsistem "*External Relations*" terdiri atas 2 *use case*, Sistem secara keseluruhan memiliki 6 aktor.



Gambar 2.1 *Use case diagram* untuk contoh *music programs*

Sumber: [SUS-07]

2.3.1.2 *Class Diagram (statis)*

Class Diagram digunakan untuk menguraikan sistem struktur *static*. Yang menunjukan kelas *object*, atribut, *methods*, dan bermacam tipe hubungan. Kelas diagram digunakan dalam seluruh metodologi *OO Development*. Kelas Dibagi dalam tiga bagian : Bagian atas – Nama Kelas, Bagian Tengah – Atribut Kelas, Bagian Bawah – Fungsi Kelas.Hubungan antar kelas ditunjukan dengan garis penghubung. Tabel 2.1 dan 2.1 menunjukan notasi UML [RAH-09].

Tabel 2.3 Komponen Kelas Diagram

Sumber: [RAH-09]

Tabel 2.4 *Multiplicity Constraints* dalam *UML*

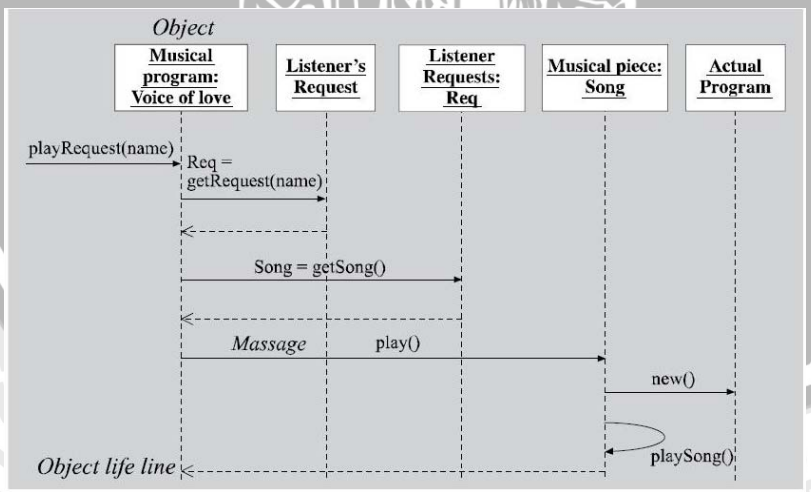
Type	Notation	Example	Meaning
Exactly 1	1 or nothing	Teacher — teaches — 1 — Course	Guru hanya mengajar satu kursus - tidak ada lagi.
0 or 1	0..1	Teacher — teaches — 0..1 — Course	Guru hanya mengajar satu kursus atau tidak sama sekali
0 or more	0..* or *	Teacher — teaches — 0..* — Course Teacher — teaches — * — Course	Guru mengajar banyak kursus atau tidak sama sekali.
1 or more	1..*	Teacher — teaches — 1..* — Course	Guru mengajar banyak kursus tetatapi minimal 1.
A defined range of numbers	3..5	Teacher — teaches — 3..5 — Course	Guru hanya mengajar antara 3 sampai 5 kursus

Sumber: [RAH-09]

2.3.1.3 *Sequence Diagram (dinamis)*

Sequence diagram menggambarkan sebuah skenario interaksi antar objek dalam *use case*. Interaksinya digambarkan oleh *message* yang terkirim dari 1 objek ke objek lainnya. *Sequence diagram* bisa digunakan sebagai cara untuk mengetahui metode apa yang perlu dilampirkan kelas-kelas objek. *Sequence Diagram* bisa digunakan untuk menggambarkan interaksi antara sistem dan elemen external dalam kasus ini disebut dengan "*Black Box Diagram*", dan ini melengkapi diagram *use case*. Diagram ini menggambarkan, setiap *use case*-nya, *subevent* yang dibuat, permintaan *event* dan reaksi sistem yang mungkin terjadi. Satu-satunya komponen dalam diagram adalah elemen eksternal dan sistem.

Penggunaan *sequence diagram* yang lain adalah untuk menggambarkan interaksi antara objek sistem dan *message* yang ditukar. *Message-message* tersebut diminta secara kronologis. Setiap *object* yang ada memiliki "*Life line*" (ditunjukan dengan garis putus-putus dibawah kotak yang menandakan *object*) Menandakan bahwa objek sedang aktif selama *event* berjalan. *Message* sebelumnya ditunjukan dengan tanda panah, dimana terhubung dengan garis objek lain. Apabila *message* sudah terkirim, berarti objek yang sudah di terima memiliki fungsi yang penting untuk melakukan layanan kebutuhan [SUS-07].



Gambar 2.2 *Sequence diagram* dari penanganan *request Music*

Sumber: [SUS-07]

2.3.1.4 *StateChart Diagram (dinamis)*

State chart menggambarkan perilaku dinamis dari suatu objek termasuk masa hidup dalam sistem. Sebuah objek memungkinkan didalam *state* yang berbeda dari masa hidupnya, tergantung dari macam-macam *events* yang terjadi dan perubahan *state* yang terjadi sekarang. *State chart* menunjukan sebuah objek yang mungkin pada *state*. Dihubungkan dengan peralihan *link* yang diketahui dari *event* yang mungkin terjadi dan oleh karena itu objek *state* berubah. *State chart* juga bisa digunakan untuk *model interface*, *control* dan *reactive system*. Digunakan utama untuk model perilaku dari objek yang memiliki *real time system*, di dalam *real time event* memiliki efek di *state* objek [WAH-05].

Tabel 2.5 State Chart Komponen

Notation	Meaning
State Name	**State :** Menunjukan state dari objek. Nama menunjukan state yang dituliskan didalam kotak.
when: Event [condition] / Action	**Transition :** Menunjukan peralihan antar state. Denagn setiap peralihan dispesifikasikan sebagai: 1. Event pemacu sebagai awal dari peralihan 2. Kondisi yang membatasi constraint yang harus dipertemukan untuk memunculkan peralihan 3. Action akan di bawa keluar ketika peralihan mengambil alih.
●	**Initial State :** Menunjukan Initial state dari objek
◉	**Final State :** Menunjukan Objek State terakhir

Sumber: [WAH-05]

Object Oriented Programming memiliki 4 macam karakteristik :

- **Pembungkusan (*Encapsulation*)**

Pembungkusan (penyembunyian Informasi) berarti meninggalkan aspek eksternal dari objek yang dapat dimasup (diakses) oleh objek lain dan memfokuskan diri pada implementasi internal suatu objek. Rincian implementasi internal dari suatu objek tersembunyi dari objek-objek lain dan terpisah dari implementasi eksternal, yaitu antarmuka (*interface*) satu objek dengan objek lainnya. Oleh karena itu, implementasi internal suatu objek dapat dirubah tanpa mempengaruhi aplikasi yang menggunakannya. Pembungkusan sebenarnya tidak unik pada pemrograman berorientasi objek tapi kemampuannya untuk

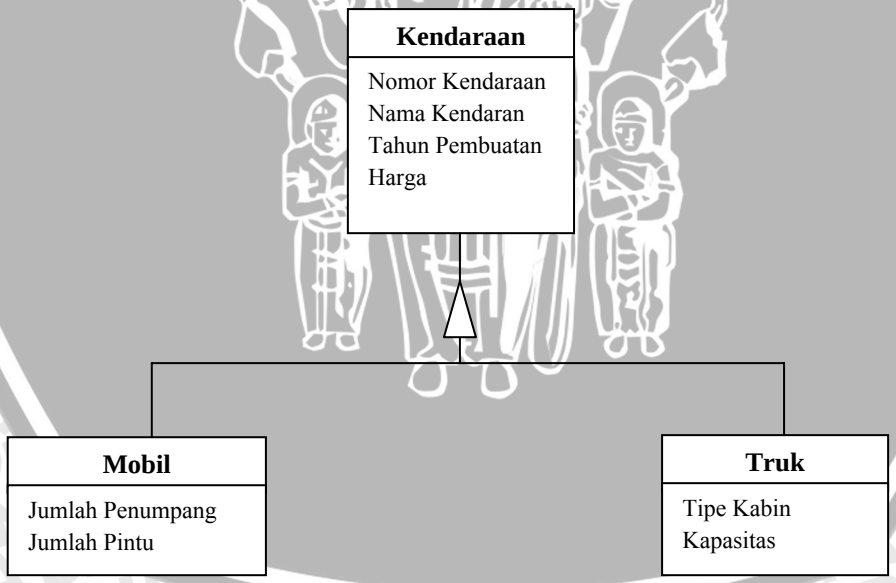
menggabungkan struktur data dan perilaku (baca: fungsi atau prosedur) dalam suatu entitas tunggal membuat bahasa berorientasi objek lebih berdaya guna dibandingkan bahasa konvensional yang memisahkan struktur data dan perilaku [PRA-04].

- **Penyembunyian Informasi (*Information Hiding*)**

Penyembunyian implementasi mengacu perlindungan implementasi internal objek. Objek disusun dari antarmuka *public* dan bagian *private* yang merupakan kombinasi data dan *method internal*. Manfaat utama adalah bagian internal dapat berubah tanpa mempengaruhi bagian-bagian program yang lain [PRA-04].

- **Generalisasi dan Pewarisan (*Inheritance*)**

Generalisasi serta Pewarisan adalah suatu cara yang sangat berdayaguna untuk berbagi apa yang dimiliki suatu kelas (atau objek) bagi kelas-kelas (atau objek-objek) yang lain. Misalkan, kita ambil contoh kelas kendaraan bermotor. Mobil, truk, dan lain-lain bisa berbagi atribut yang sama, misalnya : atribut model, tahun pembuatan, jumlah gigi transmisi, dan sebagainya.



Gambar 2.3 Pewarisan Atribut dari Kelas Pendahulu ke Kelas Turunan

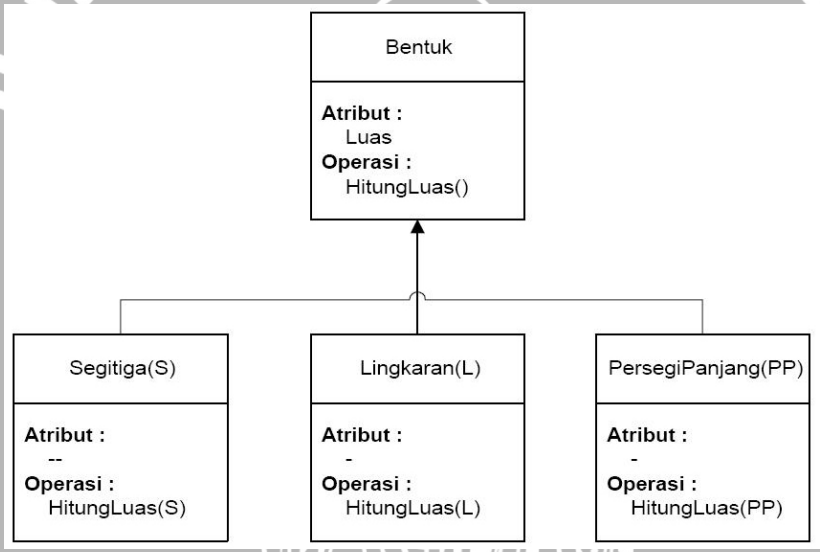
Sumber: [PRA-04]

Dari keterangan gambar diatas dapat dijelaskan bahwa generalisasi adalah proses *"bottom-up"* ("bawah ke atas") sedangkan spesialisasi adalah proses *"top-*

bottom" ("atas ke bawah") dimana pewarisan adalah berbagi atribut yang sama diantara dua kelas dibawahnya (mobil dan truk) dalam kelas yang lebih atas (kendaraan). Perlu diperhatikan bahwa pewarisan memungkinkan atribut-atribut yang cukup dituliskan sekali saja pada superkelas dan tidak perlu ditulis ulang pada subkelas yang mewarisi atribut-atribut yang sama itu [PRA-04].

- **Polimorfisme (*Polymorphism*)**

Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu *interface* yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu *interface* banyak aksi.



Gambar 2.4 Pewarisan dari Atribut yang memiliki Kelas berbeda

Sumber: [PRA-04]

Pada contoh diatas *class* dasar adalah *class* bentuk yang memiliki atribut berupa Luas dan operasi hitung luas, *class* tersebut dapat diturunkan kedalam berbagai macam *class* bentuk seperti Segitiga, Lingkaran, Persegi panjang. *Class* Segitiga, Lingkaran, Persegi panjang memiliki atribut "Luas" dari hasil penurunan *class* bentuk, akan tetapi operasi hitung luas pada masing masing *class* akan berbeda-beda, inilah yang disebut sebagai *polymorphism* [PRA-04].

2.3.2 *Entity-Relationship Diagram* (Diagram E-R)

Diagram *Entity-Relationship* menjelaskan data-data yang ada pada dunia nyata. Diagram *Entity-Relationship* menterjemahkan atau mentransformasikan data dengan memanfaatkan sejumlah model konseptual [SUY-04]. Komponen-komponen pembentuk diagram *Entity-Relationship*, antara lain:

1. Entitas dan Himpunan Entitas

   Entitas merupakan individu yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Sekelompok entitas yang sejenis dan berada dalam lingkup yang sama membentuk sebuah himpunan entitas. Entitas menunjuk pada individu suatu objek, sedangkan himpunan entitas menunjuk pada rumpun dari individu tersebut. Himpunan entitas digambarkan dengan persegi panjang dalam diagram *Entity-Relationship*.

2. Atribut

   Entitas pasti memiliki atribut yang mendeskripsikan karakteristik dari entitas tersebut. Penentuan atau pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lain dalam pembentukan model data. Atribut digambarkan dengan elips dalam diagram *Entity-Relationship*.

3. Relasi dan Himpunan Relasi

   Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Kumpulan semua relasi di antara entitas-entitas yang terdapat pada himpunan entitas-himpunan entitas tersebut membentuk suatu himpunan relasi. Himpunan relasi digambarkan dengan belah ketupat dalam diagram *Entity-Relationship*.

4. Kardinalitas atau Derajat Relasi

   Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari himpunan entitas yang satu ke himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas (misal A dan B) dapat berupa:

- Satu ke Satu (*One to One*)

  Entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B dan sebaliknya.
- Satu ke Banyak (*One to Many*)

  Entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B tetapi tidak sebaliknya. Setiap entitas pada himpunan entitas B berhubungan paling banyak dengan satu entitas pada himpunan entitas A.
- Banyak ke Satu (*Many to One*)

  Entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B tetapi tidak sebaliknya. Setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.
- Banyak ke Banyak (*Many to Many*)

  Entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B dan sebaliknya.

Diagram E-R selalu dibuat bertahap. Tahapan pertama yang dapat dilakukan adalah dengan membuat diagram E-R awal. Tujuan dari pentahapan ini adalah untuk mendapatkan sebuah rancangan basis data minimal yang dapat mengakomodasi kebutuhan penyimpanan data terhadap sistem yang sedang ditinjau. Langkah-langkah yang harus dilakukan antara lain [IRM-03]:

1. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat.
2. Menentukan atribut-atribut *key* dari masing-masing himpunan entitas.
3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas-himpunan entitas yang ada beserta *foreign key*-nya.
4. Menentukan derajat atau kardinalitas relasi untuk setiap himpunan relasi.
5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif (*non-key*).

2.4 Teori Dasar *Web*

Penyusunan skripsi ini memerlukan teori dasar *web* sebagai dasar teori yang akan memberikan penjelasan tentang HTML (*Hyper Text Markup Language*), *Hypertext Preeprocessor* (PHP), HTTPS (*HyperText Transport Protocol Secure*), *Apache Web Server*, dan Macromedia Dreamweaver MX 2004.

2.4.1 HTML (*Hyper Text Markup Language*)

HTML digunakan untuk membangun suatu halaman *web*. HTML sebenarnya bukan merupakan bahasa pemrograman, karena seperti namanya, HTML merupakan suatu bahasa *Markup* (penandaan). Tanda tersebut digunakan untuk menentukan format atau *style* dari teks yang ditandai.

Dokumen HTML diawali dengan penulisan *tag* <HTML> dan diakhiri dengan *tag* </HTML>. Sedangkan *tag* <!...> pada struktur HTML menyatakan suatu keterangan yang tidak akan diproses. *Tag* ini diletakkan baik pada HEAD maupun BODY dalam dokumen HTML. Struktur dasar dari dokumen HTML adalah sebagai berikut [SUT-07]:

```
<HTML>
        <HEAD>
                <! Bagian Kepala dari dokumen HTML>
        <\HEAD>
```

2.4.2 *Hypertext Preeprocessor* (PHP)

PHP (dulu *Personal Home Page*, sekarang *PHP : Hypertext Preeprocessor*) merupakan program yang dikembangkan secara bersama oleh para programmer dariseluruh dunia yang menekuni dunia *open-source*. PHP dikembangkan khususnya untuk mengakses dan memanipulasi data yang ada di database *server open-source* seperti MySQL. Dengan demikian, tingkat kompatibilitasnya terhadap *database server* gratis seperti MySQL sangat baik [NUG-08].

2.4.2.1 Konsep Kerja PHP

Model kerja PHP diawali dengan permintaan suatu halaman *web* oleh *browser*. Berdasarkan URL (*Uniform Resource Locator*) atau dikenal dengan sebutan alamat internet, *browser* mendapatkan alamat dari *web server*,

mengidentifikasi halaman yang dikehendaki dan menyampaikan segala informasi yang dibutuhkan oleh *web server*. Selanjutnya, *web server* akan mencarikan berkas yang diminta dan apabila sudah didapatkan maka *web server* akan mengirimkannya ke mesin PHP. Mesin inilah yang kemudian akan memproses dan memberikan hasilnya (berupa kode HTML) ke *web server*. Selanjutnya *web server* akan menyampaikan hasil tersebut ke *client*. Skema konsep kerja PHP ditunjukkan dalam Gambar 2.5 [WAH-05].

2.4.2.2 Variabel Pada PHP

Setiap variabel dalam PHP selalu dimulai dengan tanda dolar ($) dan harus dimulai dengan huruf atau garis bawah (_) dan kemudian dapat diikuti oleh huruf, angka ataupun garis bawah. `$warpspeed`, `$impuls_speed`, `$LCAR` dan `$Dilithium1` adalah contoh penamaan variabel PHP yang benar.

Jenis suatu variabel pada PHP ditentukan pada saat jalannya program (*runtime*). PHP tidak memerlukan pendeklarasian variabel terlebih dahulu karena PHP mempunyai kepandaian untuk membedakan jenis variabel secara otomatis berdasarkan konteks yang sedang berlaku bagi variabel tersebut dan tipe datanya bisa diubah sesuai dengan keinginan [IRA-03].

PHP mendukung beberapa jenis variabel sebagai berikut:

1. *Integer*

   Variabel berjenis *integer* bertujuan untuk menyimpan bilangan bulat.

2. *Double*

   Untuk menyimpan bilangan bernilai pecahan dan juga bilangan pemangkatan.

3. *String*

   String merupakan jenis data karakter yang disimpan sebagai nomor pada memori komputer. Nilai yang disimpan adalah nilai ASCII karakter *string* tersebut.

4. *Array*

   Array adalah sebuah set variabel yang mempunyai jenis data sama. *Array* mengandung komponen yang disebut elemen dan disimpan pada lokasi tertentu pada memori.

5. *Object*

   Jenis variabel objek adalah berdasarkan gambaran objek pada dunia nyata yang mempunyai status dan tingkah laku. Sebuah variabel objek menyimpan statusnya dalam bentuk variabel dan tingkah lakunya dalam bentuk parameter.

2.4.2.3 *Script* PHP

Terdapat dua cara yang sering digunakan untuk menulis *script* PHP, yaitu [PRA-05]:

1. *Embedded Script*

   Cara ini dilakukan dengan meletakkan *script* PHP di antara *tag-tag* HTML. Berikut contoh penggunaannya:

```
<html>
<head>
<title> embedded script </title>
</head>
<body>
<? echo "Ini adalah contoh embedded script"; ?>
</body>
</html>
```

2. *Non-Embedded Script*

   Non-embedded script merupakan pembuatan program murni PHP, dimana *tag-tag* HTML yang diletakkan didalamnya. Berikut contoh penggunaannya:

```
<?
   echo"<html>";
   echo"<head>";
   echo"<title> non embedded script </title>";
   echo"</head>";
   echo"<body>";
   echo"Ini adalah contoh non embedded script";
   echo"</body>";
   echo"</html>";
?>
```

2.4.2.4 *Tag* PHP

Penulisan program PHP dapat dilakukan dengan menggunakan berbagai *tag.* Hal ini tidak akan mempengaruhi hasil program yang dibuat [PUT-06]. *Tag* yang dapat digunakan adalah sebagai berikut:

1. *Style* Standar

   Style standar PHP sangat mirip dengan penulisan program XML, yakni diawali dengan **<?php** dan diakhiri dengan **?>**. Contoh penulisannya adalah

```
<html>
<head>
<title> style standar </title>
</head>
<body>
<?php echo "Ini adalah contoh style standar PHP"; ?>
</body>
</html>
```

2. *Short Style*

   Penulisan *script* pada *style* ini cukup praktis bila dibandingkan dengan *style* sebelumnya karena hanya diawali dengan **<?** dan diakhiri dengan **?>**. Contoh penulisannya adalah:

```
<html>
<head>
<title> short style </title>
</head>
<body>
<? echo "Ini adalah contoh short style PHP"; ?>
</body>
</html>
```

3. *Style* JavaScript

   Style ini digunakan apabila telah terbiasa menggunakan pemrograman JavaScript. Penulisan programnya diawali dengan **<SCRIPT LANGUAGE =**

'PHP'> dan diakhiri dengan **</SCRIPT>.** Contoh penulisannya adalah:

```
<html>
<head>
<title> style JavaScript </title>
</head>
<body>
<SCRIPT LANGUAGE = 'PHP'>
     echo "Ini adalah contoh style Javascript";
</SCRIPT>
</body>
</html>
```

4. *Style* ASP

Diawali dengan **<%** dan diakhiri dengan **%>**. Contoh penulisannya adalah:

```
<html>
<head>
<title> style ASP </title>
</head>
<body>
<% echo "Ini adalah contoh style ASP"; %>
</body>
</html>
```

2.4.2.5 *Cookie*

Cookie adalah sepotong data yang disimpan pada *harddisk* lokal milik pengunjung (*client*) dan digunakan oleh halaman *web* dalam mengingat sesuatu informasi. Data pada *harddisk* lokal ini dikenali oleh *server* untuk mendapatkan kembali informasi yang pernah dikirimkan ke *client*.

Cookie bersifat sementara, artinya ketika *browser* ditutup atau waktu penyimpanan *cookie* tersebut telah habis. *Cookie* juga akan secara otomatis dihapus. Dengan demikian, ketika *browser* kembali dibuka atau melewati jangka waktu tertentu, maka nilai *cookie* tersebut akan kosong. Sebuah *client* hanya dapat memegang 300 *cookie* pada satu saat dan sebuah *server* hanya dapat mengirim 20 *cookie* ke sebuah *client*. Sebuah *cookie* hanya bisa berukuran sampai dengan 4 kilo*byte*, sehingga total memori *harddisk* yang digunakan *cookie* hanya mencapai 1,2 mega*byte* [WAH-05].

Web server mengirim *cookie* ke *client* melalui judul HTTP, yang dikirim sebelum teks HTML. Melalui judul pada HTTP inilah *client* bisa mengetahui apakah ia perlu mengirimkan *cookie* yang ada pada *harddisk* lokal ke *web server* atau tidak. Seperti halnya dengan *web server*, *client* juga mengirimkan kembali *cookie* ke *web server* melalui judul HTTP. Di dalam judul HTTP, *cookie* diatur melalui set-cookie. set-cookie mempunyai informasi nama *cookie* dan nilainya, batas kadaluarsa, *path*, *domain* dan parameter keamanan. Informasi nama variabel dan nilai dari *cookie* ditunjukkan dalam Tabel 2.6.

Tabel 2.6 Parameter *cookie*

Informasi	Keterangan
Path	Digabungkan dengan *domain*, nilai *path* menentukan direktori pada *web server* yang dapat menggunakan cookie. Jika informasi *path* dan URL yang diminta tidak cocok, maka client tidak akan mengirimkan *cookie*. Nilai bawaan pada path berupa "/" yang berarti bahwa *cookie* valid untuk semua direktori pada *web server*.
Domain	Menyatakan *domain* (alamat) *server* yang mendefinisikan *cookie*. *Client* tidak akan mengirimkan *cookie* apabila *path* dan URL yang diminta tidak cocok dengan *domain* ini. Salah satu pemanfaatannya adalah untuk berbagi *cookie* pada beberapa *server*.
Expires	Menyatakan batas waktu kadaluarsa. Bawaannya, *cookie* hanya berlaku sampai *browser* ditutup.
Secure	Untuk menentukan pengiriman *cookie* hanya apabila protokol HTTPS (HTTP *Secure*) digunakan.

Sumber : [KUR-06]

2.4.2.6 *Session*

Session dapat digunakan untuk membentuk interaksi antara sebuah *client* dengan *web server* dalam selang waktu tertentu. PHP menyediakan pustaka yang berguna untuk membentuk sebuah *session* dengan menggunakan *session*. Berbeda dengan *cookie* yang menyimpan data pada *client*, *session* diimplementasikan dengan menyimpan data pada *server*. Dengan demikian tidak perlu ada komunikasi bolak-balik antara *web server* dan *client* ketika *web server* membutuhkan data tersebut.

Session akan dimulai ketika *client* mulai masuk sebuah situs dan akan berakhir begitu *client* tersebut menutup halaman situs yang telah dibukanya. *Client* akan mendapat variabel yang terus ada selama ia melakukan kunjungannya tersebut. Setiap kali suatu *session* dibentuk, maka akan terdapat referensi yang menunjuk ke *session* yang bersangkutan. Referensi ini dikenal dengan SID (*Session Identifer*). SID ini akan disimpan sebagai *cookie* apabila fasilitas *cookie* dalam keadaan dihidupkan pada *browser*, sedangkan data *session*-nya akan disimpan di *server* [WAH-05].

2.4.2.7 Keunggulan PHP

Script PHP ini merupakan saingan berat ASP. Pada dasarnya memang cara kerja kedua bahasa pemrograman *web* tersebut memilki persamaan, yaitu script disisipkan pada HTML dan dijalankan oleh *web server*. Sebagai sebuah bahasa pemrograman *server-side*, PHP juga memiliki keunggulan seperti [NUG-08]:

- *Source* program atau *script* tidak dapat dilihat menggunakan fasilitas *view* HTML *source*, yang ada pada *web browser*, seperti Internet Explorer atau semacamnya.
- *Script* tersebut dapat memanfaatkan sumber-sumber aplikasi yang dimiliki oleh *server*, seperti misalnya untuk keperluan *database connection*. Saat ini PHP sudah mampu melakukan koneksi dengan berbagai database seperti MySQL, DirectMS-SQL, Velocis, IBM DB2, Interbase, PostgreSQL, dBase, FrontBase, Solid, Empress, mSQL, Sybase, FilePro (read-only-Personix, Inc.), Unix dbm, informix dan bahkan semua

database yang mempunyai *provider* ODBC, seperti misalnya Microsoft Acces dan lain-lain.

- Pada aplikasi yang dibuat dengan PHP, saat dijalankan *server* akan mengerjakan *script* dan hasilnya dikirim ke *web browser*. Hal itu menyebabkan aplikasi tidak memerlukan kompatibilitas *web browser* atau harus menggunakan *web browser* tertentu dan pasti dikenal oleh *web browser* apapun.
- PHP dapat melakukan semua aplikasi program CGI, seperti mengambil nilai *form*, menghasilkan halaman *web* yang dinamis, mengirimkan dan menerima *cookie*. PHP juga dapat berkomunikasi dengan layanan yang menggunakan protokol IMAP, SNMP, NNTP, POP3, HTTP dan lainnya.

2.4.3 *Apache Web Server*

Untuk membuat sebuah pemrograman *web* dinamis, diperlukan sebuah *web server*. Ada banyak *web server* yang berkembang dan sering digunakan dalam membangun aplikasi berbasis *web*, salah satunya adalah Apache. Apache memiliki beberapa kelebihan antara lain adalah [GAL-07]:

- *Free of Charge*, berarti tidak harus membayar lisensi kepada pembuat untuk menggunakannya.
- Dapat diakses (API ke berbagai *scripting language*) dan digabung dengan berbagai aplikasi lain (databaseserver, ssl, ext) dan sebagainya.
- Waktu pemrosesan lebih cepat dan tangguh denga konfigurasi yang benar.
- Dapat dilakukan setting dan instalasi sesuai dengan kebutuhan dengan adanya *modules* dan DSO-nya.
- Memiliki kemampuan *advanched setting* dan *configuration support*.

Dengan berbagai keunggulan tersebut, Apache sangat bagus jika dikombinasikan dengan aplikasi lainnya. Penggabungan yang paling sering adalah dengan menggabungkan Apache, PHP dan MySQL yang berjalan di *server* Linux atau yang terkenal dengan dengan LAMP (Linux, Apache, Mysql, PHP). Sementara bagi pengguna windows Apache, PHP dan MySQL juga bisa diinstal pada OS tersebut [SAL-07].

2.5 Jaringan Komputer

Jaringan komputer bukanlah sesuatu yang baru saat ini. Hampir di setiap perusahaan terdapat jaringan komputer untuk memperlancar arus informasi di dalam perusahaan tersebut. Internet yang mulai populer saat ini adalah suatu jaringan komputer raksasa yang merupakan jaringan komputer yang terhubung dan dapat saling berinteraksi. Hal ini dapat terjadi karena adanya perkembangan teknologi jaringan yang sangat pesat, sehingga dalam beberapa tahun saja jumlah pengguna jaringan komputer yang tergabung dalam Internet berlipat ganda.

2.5.1 Definisi Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan komputer, printer dan peralatan lainnya yang terhubung. Informasi dan data bergerak melalui kabel-kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada printer yang sama dan bersama sama menggunakan hardware/software yang terhubung dengan jaringan. Tiap komputer, printer atau periferal yang terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan node. Sebuah jaringan biasanya terdiri dari 2 atau lebih komputer yang saling berhubungan diantara satu dengan yang lain, dan saling berbagi sumber daya misalnya CDROM, Printer, pertukaran *file*, atau memungkinkan untuk saling berkomunikasi secara elektronik. Komputer yang terhubung tersebut, dimungkinkan berhubungan dengan media kabel, saluran telepon, gelombang radio, satelit, atau sinar infra merah [SYA-05].

2.5.2 Keuntungan Penggunaan Jaringan Komputer

Keuntungan yang diperoleh dari penerapan teknologi jaringan adalah sebagai berikut [SUT-05]:

1. *Resource sharing*, yaitu dapat berbagi sumber daya. Misal, pemakaian satu *printer* untuk beberapa komputer yang terhubung dalam jaringan.
2. *File sharing*, antar komputer dapat melakukan pertukaran data atau *file*.

3. Reliabilitas tinggi, dengan menggunakan jaringan komputer maka akan memiliki sumber-sumber alternatif. Misal, semua *file* dapat disimpan atau di-*copy* dalam dua, tiga atau lebih komputer yang terhubung dalam jaringan. Sehingga apabila salah satu mesin mengalami kerusakan, maka masih ada salinan yang bisa digunakan di tempat lain.
4. Menghemat biaya, penghematan biaya terjadi karena komputer berukuran kecil/*PC* mempunyai rasio harga /kinerja yang lebih baik dibandingkan dengan komputer besar. Komputer besar seperti *mainframe* memiliki kecepatan sekitar sepuluh kali lipat kecepatan komputer kecil/*PC*. Tetapi harga sebuah *mainframe* bisa ribuan kali lebih mahal dibanding *PC*.
5. Kemudahan komunikasi, komunikasi antar komputer dalam suatu lingkungan kerja dapat dilakukan dengan mudah, misal dengan adanya program *E-mail* atau *Chatting*.
6. Apabila salah satu unit komputer terhubung ke internet melalui *modem* atau LAN, maka semua atau sebagian unit komputer pada jaringan juga dapat mengakses internet dengan metode *sharing connection*.
7. Fasilitas *mapping*, *mapping* berfungsi untuk memetakan suatu *directory* pada *server/workstation* yang terhubung dalam jaringan sedemikian sehingga *directory* tersebut seolah-olah menjadi *drive* lokal. *Mapping* hanya bisa dilakukan apabila komputer sumber dan komputer tujuan terhubung melalui jaringan, dan *directory* pada komputer sumber berada pada status *sharing*.

2.5.3 Tipe Jaringan Komputer

Berdasarkan luasnya jangkauan, jaringan komputer dapat dibedakan menjadi empat kelompok [SUT-05]:

1. *Workgroup*

   Tipe jaringan *workgroup* merupakan jaringan yang menghubungkan sejumlah terbatas komputer dalam sebuah ruangan (misal dalam kampus). Tipe ini biasanya dimiliki oleh sebuah institusi/perusahaan/lembaga dan dioperasikan secara mandiri.

2. *Local Area Network/LAN*, yaitu suatu jaringan komunikasi data yang luas jangkauannya meliputi suatu area lokal tertentu. Misal jaringan komunikasi data di suatu gedung. Sebagaimana tipe *workgroup*, *LAN* biasanya dimiliki oleh perusahaan/instansi/lembaga dan dioperasikan secara mandiri.
3. *Metropolitan Area Network/MAN*, yaitu suatu jaringan komunikasi data yang luas jangkauannya meliputi area dalam satu kota, misal jaringan komputer di kota Yogyakarta. *MAN* bisa terbentuk oleh gabungan/hubungan beberapa *LAN*.
4. *Wide Area Network/WAN*, yaitu suatu jaringan komunikasi data yang luas jangkauannya meliputi antar kota atau antar negara, missal jaringan komunikasi data pada *Internet*. *WAN* terbentuk oleh dua atau lebih jaringan yang dihubungkan melalui *router*. *WAN* menggunakan media komunikasi publik.

2.5.4 Internet

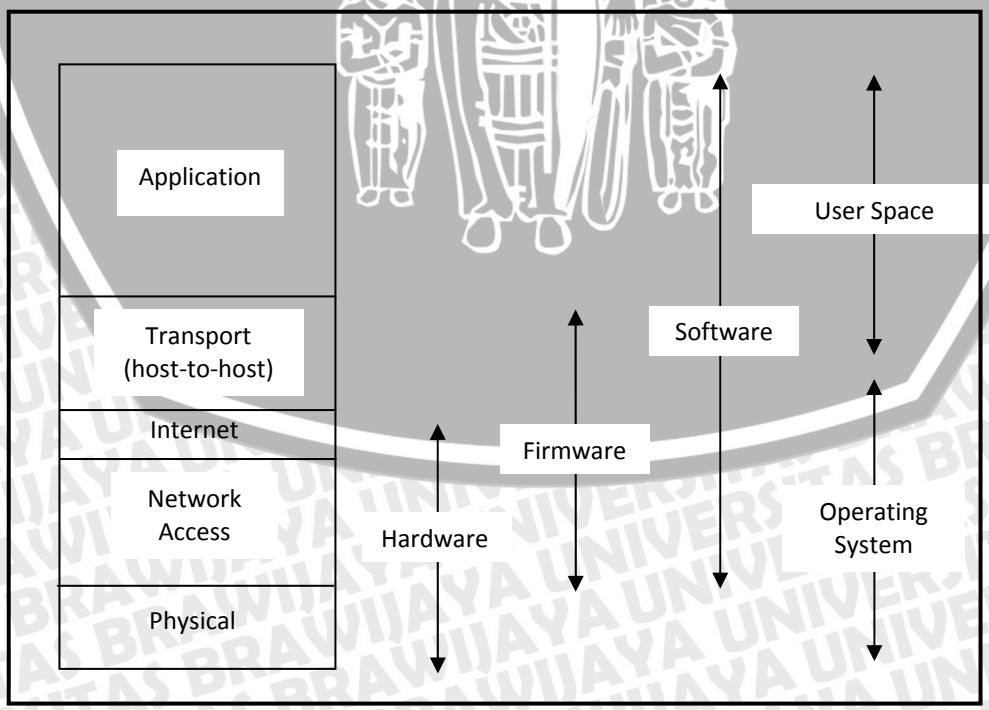
Interconnected Network atau yang lebih populer dengan sebutan *Internet*, adalah sebuah sistem komunikasi global yang menghubungkan komputer-komputer dan jaringan-jaringan komputer di seluruh dunia. Setiap komputer dan jaringan terhubung - secara langsung maupun tidak langsung - ke beberapa jalur utama yang disebut *internet backbone* dan dibedakan satu dengan yang lainnya menggunakan *unique name* yang biasa disebut dengan alamat IP 32 bit. Contoh: 202.155.4.230. Komputer dan jaringan dengan berbagai *platform* yang mempunyai perbedaan dan ciri khas masing-masing bertukar informasi dengan sebuah protokol standar yang dikenal dengan nama TCP/IP (*Transmission Control Protocol/Internet Protocol*). TCP/IP tersusun atas 4 layer [SUT-03]:

1. *Layer network* bertanggung jawab mengirim dan menerima data ke dan dari media fisik. Media fisiknya dapat berupa kabel, serat optik atau gelombang radio. Mampu menterjemahkan sinyal listrik menjadi data digital yang di mengerti oleh komputer, yang berasal dari peralatan lain yang sejenis.

2. *Layer internet* bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. Pada *layer* ini terdapat tiga macam protokol, yaitu IP, ARP, dan ICMP. IP (*Internet Protocol*) berfungsi untuk menyampaikan paket data ke alamat yang tepat. ARP (*Address Resulotion Protocol*) ialah protokol yang digunakan untuk menemukan alamat *hardware* dari *host*/komputer yang terletak pada *network* yang sama. Sedangkan ICMP (*Internet Control Massage Protocol*) ialah protokol yang digunakan untuk mengirimkan pesan dan melaporkan kegagalan pengiriman data.

3. *Layer transport* berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua *host*/komputer. Pada lapisan *Transport* menggunakan *Acknowledgement* positif dan *Acknowledgement* negatif pada aliran datanya. *Acknowledgment* positif akan memberitahukan pesan apabila data yang di transferkan telah sampai sedangkan *Acknowledgement* negatif jika paket yang ditransfer tidak sampai ke tujuan maka akan terjadi pengiriman ulang. Kedua protokol tersebut ialah TCP (*Transmission Control Protokol*) dan UDP (*User Datagram Protocol*).

4. *Layer application*. Pada layer ini terletak semua aplikasi yang menggunakan protokol TCP/IP misalnya http, ftp, telnet, smpt dan lain sebagainya.



Gambar 2.5 Lapisan Protokol TCP/IP

Sumber: [SUT-04]

2.5.5 Topologi Jaringan

Topologi menggambarkan struktur jaringan, atau begaimana sebuah jaringan didesain. Terdapat dua definisi topologi. *Physical opology*, merupakan *layout* aktual dari kabel-kabel (media) jaringan. Dan, *logical topology*, mendefinisikan bagaimana media diakses oleh *host-host*. Adapun topologi fisik yang umum digunakan dalam membangun sebuah jaringan adalah [SYA-05]:

- **Topologi bus**

  Merupakan segmen *backbone* tunggal melalui kabel lurus panjang, di mana semua *host* dikoneksikan langsung

- **Topologi bintang**

  Menghubungkan semua kabel ke sebuah poin sentral. Poin ini biasanya berupa sebuah *hub* atau *switch.*

- **Topologi cincin**

  Mengkoneksikan *host* pertama ke *host* berikutnya, dan *host* terakhir ke *host* pertama. Model ini akan membuat lingkaran *node-node* komputer yang dikoneksikan melalui media kabel.

- **Topologi *mesh***

  Digunakan pada kondisi di mana tidak ada hubungan komunikasi terputus secara absolut antar *node* komputer. Sebagai contoh adalah sistem-sistem kontrol dari sebuah *nuclear power plant*. Topologi ini merefleksikan juga bagaimana desain dari internet, yang memiliki *multi path* ke berbagai lokasi.

- **Topologi pohon**

  Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai hirarki semakin

tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer .

Sedangkan *logical topology* dari sebuah jaringan menggambarkan bagaimana *host-host* saling berkomunikasi melalui sebuah media. Dua tipe umum dari *logical topology* adalah [ZUL-07]:

- ***Broadcast topology***

  Pada topologi model ini, setiap host mengirim datanya ke semua *host* lain dalam media jaringan, dan di sana tidak ada *station* pesanan apapun dalam *network*, yang pertama dating, pertama dilayani. Konsep ini berlaku juga pada cara kerja *Ethernet*.

- ***Token passing***

  Token passing mengontrol akses dengan melepas pesan elektronik ke setiap *host* secara berurutan. Saat sebuah *host* menerimanya, ini memberi arti bahwa *host* tersebut dapat mengirim data dalam *network*. Jika *host* tidak memiliki data untuk dikirim, ia melepas pesan ke *host* berikutnya dan proses similar ini akan berulang-ulang.

BAB III

METODOLOGI

Metodologi penelitian menjelaskan mengenai langkah-langkah yang dilakukan untuk merealisasikan aplikasi perangkat lunak yang akan dibuat. Langkah-langkah yang diperlukan antara lain studi literatur, survei dan pengumpulan data alumni, perencanaan dan pembuatan formulir, pembuatan *website*, pengolahan dan analisa data, pengujian aplikasi serta pengambilan kesimpulan dan saran.

3.1 Studi Literatur tentang Sistem Survei, Pengolahan, dan Penyimpanan Data Alumni.

Langkah pertama yang dilakukan adalah melakukan studi literatur dan mempelajari sistem penyimpanan data alumni dan analisanya dari beberapa buku referensi. Dari hasil survei maka akan dapat dianalisa kelebihan dan kekurangan sistem survei dan penyimpanan data sehingga dapat dipilih sistem yang sesuai dengan kondisi alumni UB.

3.2 Pengumpulan data alumni.

Langkah selanjutnya yang akan dilakukan dalam penelitian ini adalah mengumpulkan data-data tentang alumni. Pengumpulan data akan dilakukan dengan mengambil data alumni yang tersimpan dalam database alumni masing-masing Fakultas dan Jurusan.

Selain itu, data alumni bisa juga diperoleh melalui internet. Caranya yaitu dengan memasang *website* di internet, kemudian kita hubungi alumni melalui *e-mail* agar mereka membuka *website* tersebut. Biasanya alumni tiap-tiap angkatan dan jurusan sudah membuat *mailing-list* sehingga tinggal dikirimkan *e-mail* ke *mailing-list* tersebut.

3.3 Perencanaan dan Pembuatan Formulir *Tracer Study*.

Sebelum formulir data alumni dibuat, terlebih dahulu ditentukan/dipilih data-data yang diperlukan. Pemilihan data ini didasarkan pada kebutuhan akan

data alumni dalam rangka pengembangan UB. Kebutuhan data alumni ini mengacu pada program-program rencana ataupun proposal yang sering dibuat untuk keperluan pengembangan masing-masing jurusan. Setelah kebutuhan akan data alumni ditentukan, selanjutnya dibuat formulir yang diperlukan.

3.4 Pembuatan *Website*.

Selain dengan menyebarkan formulir, pengumpulan data juga akan dilakukan dengan membuat *website* yang di dalamnya berisi formulir yang bisa diisi oleh pengunjung yang merupakan alumni UB. Alumni dikirimi *e-mail* agar membuka *website* tersebut dan mengisi formulir yang disediakan.

3.5 Pengolahan dan Analisa Data Alumni

Setiap data yang diperoleh selanjutnya dianalisa alumni yang bersangkutan meliputi: tahun masuk, tahun lulus, lama studi, indeks prestasi, aktivitas selama kuliah, lama tunggu sebelum memperoleh kerja pertamakali, bidang usaha tempat bekerja, posisi di perusahaan, gaji pertamakali kerja. Dari data yang diperoleh dapat dilihat apakah alumni UB dapat bisa cepat diserap oleh lapangan kerja, apakah bidang usaha tempat alumni bekerja sesuai dengan latar belakang pendidikan yang diperoleh, apakah alumni cukup dihargai di dunia kerja dilihat dari gaji yang mereka terima pertamakali bekerja.

3.5.1 Fitur Program

Fitur program merupakan fasilitas-fasilitas yang disediakan dalam *website tracer study* untuk *user*. Dengan adanya fitur-fitur ini maka *user* dapat dengan mudah mendapatkan informasi yang dibutuhkan dalam *web* ini.

3.5.1.1 Pengisian dan Modifikasi Data

Fasilitas untuk mengisi data dan melakukan modifikasi (edit/delete) dengan mudah dan *user friendly*. Data yang dapat disikan adalah:

- Data Alumni: Data Pribadi – Riwayat Pekerjaan – Riwayat Pendidikan
- Data Perusahaan (Nama, Alamat dan Bidang Kerja)
- Data Fakultas dan Jurusan

3.5.1.2 Pencarian dan Pengelompokan Data

Data yang dimasukkan dapat dicari dan diurutkan berdasarkan kata kunci tertentu yang diisikan oleh *user*.

Kategori pencarian adalah berdasarkan:

- *Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun Lulus.*

Sedangkan Kategori pengelompokkan data adalah berdasarkan:

- *Semua Fakultas, Per Fakultas tertentu, Semua Jurusan,* dan *Per Jurusan tertentu*

3.5.1.3 Laporan

Data yang telah masuk dapat direkap untuk membuat laporan berdasarkan pilihan laporan berikut:

1.Data Kelengkapan Studi yang meliputi :

a. Rata-rata Lama Studi
b. Rata-rata IPK

2. Data Pekerjaan Lulusan yang meliputi :

a. Rekapitulasi Masa Tunggu
b. Rekapitulasi Kriteria Perekrutan Pekerja
c. Rekapitulasi Bidang Utama Pekerjaan
d. Rekapitulasi Gaji Pertama
e. Rekapitulasi Kesesuaian Bidang Kerja
f. Rekapitulasi Lama Masa Kerja dan Masa Jabatan di Perusahaan
g. Rekapitulasi Alasan untuk Menerima Pekerjaan yang tidak sesuai dengan Studi yang diambil

Pengelompokkan laporan dapat dipilih berdasarkan:

1. Range Tahun Kelulusan
2. Range Tahun Masuk (angkatan)
3. Per Jurusan / Semua Jurusan (Fakultas)

Dan laporan dapat ditampilkan dalam format tampilan:

1. Baris / Daftar

3.5.1.4 Fasilitas Khusus

Selain itu ada fasilitas tambahan sebagai pendukung dari program aplikasi ini yaitu:

1. Pencarian data alumni yang bekerja pada perusahaan tertentu.
2. Antar anggota dapat melakukan komunikasi melalui fórum diskusi.
3. Anggota bisa memberikan *comment* kepada anggota lainnya.

4. Pencetakkan (*print*) data dari List Hasil Pencarian.
5. Pencetakkan (*print*) per detail data alumni.

3.6 Pengujian Aplikasi

Pengujian aplikasi yaitu untuk mengetahui kesesuaian analisa kebutuhan yang dibuat dengan implementasi aplikasi. Analisis sistem dilakukan dengan membandingkan sistem aplikasi dengan teori yang ada sehingga didapatkan suatu kesimpulan. Metode pengujian yang digunakan adalah metode pengujian *White Box* dan *Black box*. Pengujian dimulai dari pengujian unit, kemudian dilanjutkan dengan pengujian integrasi, dan berakhir pada validasi atau pengujian sistem.

- Pengujian klas-klas pembangun aplikasi dengan menggunakan metode *White Box* dengan teknik *Basis-Path.*
- Pengujian terintegrasi hasil implementasi aplikasi dengan mengunakan metode *Scenario-Based.*

Pengujian validasi dilakukan untuk mengetahui validitas aplikasi berdasarkan kebutuhan fungsional dan non fungsional yang ditentukan dengan meggunakan metode *Black box*.

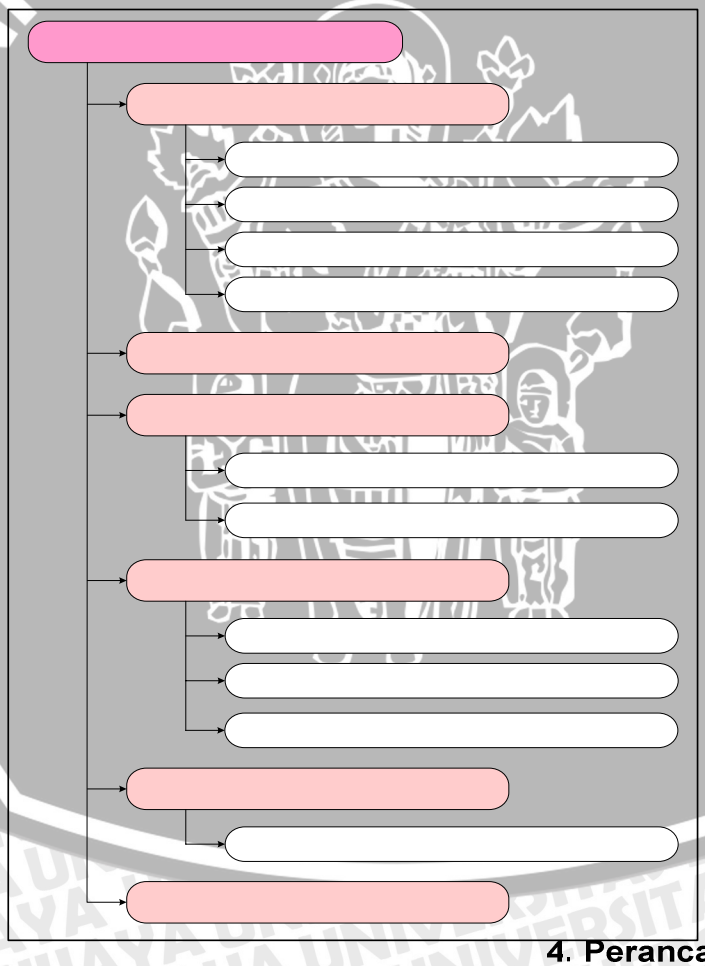
3.7 Pengambilan Kesimpulan dan Saran

Setelah mendapatkan hasil dan analisis dari pengujian *software*, maka langkah berikutnya adalah penarikan kesimpulan dari data-data yang didapat dan memberi saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi serta menyempurnakan penulisan.

BAB IV

ANALISIS KEBUTUHAN DAN PERANCANGAN

Bab ini menjelaskan analisis dan perancangan *Web* Penelusuran Data Alumni UB. Analisis dibutuhkan untuk menentukan kelompok *user* dan memaparkan kebutuhan *user* terhadap perangkat lunak yang akan dibuat sehingga perangkat lunak yang dihasilkan diharapkan dapat mengakomodasi kebutuhan semua *user*. Bab ini meliputi analisis kebutuhan, *Standard Operational Procedure* (SOP), perancangan sistem, perancangan basis data, perancangan proses dan perancangan *user interface*. Diagram alir analisis dan perancangan perangkat lunak ditunjukkan dalam gambar 4.1.



Gambar 4.1 Diagram Alir Analisis dan Perancangan Perangkat Lunak

[Analisis]

4.1 Analisis Kebutuhan

Analisis dibutuhkan untuk menjembatani antara kebutuhan perangkat lunak yang dibuat dengan perancangan, sehingga perangkat lunak yang dihasilkan dapat diimplementasikan.

Proses analisis dalam pembuatan sistem ini meliputi penjelasan cara kerja sistem yang akan dirancang dalam bentuk analisis kebutuhan perangkat lunak, analisis kebutuhan alumni dan lembaga pendidikan, kelompok pengguna,dan kebutuhan pengguna.

4.1.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak adalah aktifitas rekayasa perangkat lunak yang menjembatani antara kebutuhan ditingkat sistem dan perancangan perangkat lunak. Analisis kebutuhan adalah proses yang digunakan untuk mendapatkan, menganalisis, dan memvalidasi kebutuhan-kebutuhan sistem.

4.1.2 Analisis Kebutuhan Alumni dan Lembaga Pendidikan

Penelusuran dan pengumpulan data alumni UB selama ini lebih banyak dilakukan secara manual. Hal ini menyebabkan data yang ada tidak ter-*update* dengan baik. Oleh karena itu dibutuhkan suatu sistem yang bisa memberikan kemudahan dalam mengakses data-data tersebut. Para alumni bisa memanfaatkan system tersebut untuk meng*update* data pribadinya maupun melihat data alumni lain, sehingga bisa tercipta komunikasi yang baik antar alumni.

Bagi UB sebagai instansi pengelola, dengan adanya system terpusat penelusuran data alumni ini akan bisa memberikan manfaat berupa informasi dari para alumni, keperluan proses akreditasi, untuk menilai relevansi pendidikan tinggi, perbaikan kurikulum, ataupun hanya melihat profil alumninya. Hal-hal tersebut lah yang bisa dijadikan referensi untuk pengembangan UB di masa yang akan datang.

4.1.3 Kelompok Pengguna

Kelompok pengguna merupakan pihak-pihak yang terkait dengan penelusuran data alumni ini. Kelompok pengguna tersebut sebagai berikut :

- Administrator (PUSKOM UB)
- Alumni dan Mahasiswa Lulus
- User Eksternal

- Fakultas dan Jurusan

4.1.4 Kebutuhan Pengguna

Setelah mendefinisikan pihak-pihak yang akan menggunakan sistem, langkah selanjutnya adalah menjelaskan secara terperinci kebutuhan pengguna terhadap sistem. Penjelasan pihak-pihak yang akan menggunakan sistem ini dijelaskan pada tabel 4.1.

Tabel 4.1 Deskripsi Aktor

No.	Aktor	Penjelasan
1.	Administrator	*User* yang mempunyai hak akses tertentu terhadap sistem. Admin memerlukan sistem yang dapat membantu mempermudah administrasi data *user*. Pada *web tracer study* ini PUSKOM UB yang akan berperan sebagai admin.
2.	Alumni dan Mahasiswa Lulus	*User* ini memiliki hak tertentu untuk mengakses system. *User* ini harus melakukan registrasi terlebih dahulu sebelum melakukan pengisian data dan kuisioner, melihat informasi, maupun melakukan manipulasi data yang lain.
3.	*User* Eksternal	*User* yang tidak memiliki keanggotaan dalam *tracer study* ini. *User* ini hanya dapat melihat informasi umum dan data alumni yang tidak bersifat pribadi.
4.	Fakultas	*User* yang memiliki hak akses tertentu terhadap system. *User* ini bisa menambahkan data alumni Fakultas dan Jurusan masing-masing serta melihat data alumni secara lebih detil serta mencetaknya sebagai arsip.
5.	Jurusan	*User* yang memiliki hak akses tertentu terhadap system. *User* ini hanya bisa menambahkan data alumni Jurusannya serta

		melihat data alumni secara lebih detil.

Sumber: [Analisis]

Deskripsi aktor yang telah dibuat selanjutnya akan dirancang daftar kebutuhan yang berisi kebutuhan yang harus disediakan oleh sistem. Daftar kebutuhan ini terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem, dan pada kolom yang lain akan menunjukkan nama *use case* yang akan menyediakan fungsionalitas masing-masing kebutuhan tersebut. Daftar kebutuhan fungsional dan non fungsional sistem yang dikembangkan pada tugas akhir ini ditunjukkan pada Tabel 4.2.

Tabel 4.2 Daftar Kebutuhan Fungsional Sistem

No.	*Use Case*	Kebutuhan
1.	*Login*	Sistem harus memberikan fasilitas untuk *login*, sehingga hanya *user* tertentu yang dapat mengakses informasi.
2.	*Logout*	Sistem harus menyediakan fasilitas untuk *logout* agar *user* dapat keluar dari sistem.
3.	Menambahkan *Account*	Sistem harus menyediakan fasilitas bagi administrator untuk menambah *account* baru yang dapat melakukan akses terhadap sistem ini dengan hak-hak tertentu sesuai kapasitas dan keperluannya terhadap sistem ini.
4.	Melihat *Account*	Sistem harus menyediakan fasilitas bagi administrator untuk melihat *account-account* yang ada.
5.	Mengubah *Account*	Sistem harus menyediakan fasilitas bagi administrator untuk mengubah *account-account* yang ada

6.	Menghapus *Account*	Sistem harus menyediakan fasilitas bagi administrator untuk menghapus *account-account* yang ada
7.	Mengisi Data Registrasi	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melakukan proses pengisian data registrasi anggota.
8.	Menambah Data Profil	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menambah data pribadinya.
9.	Melihat Data Profil	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, serta fakultas dan jurusan untuk melihat data alumni.
10.	Mengganti Data Profil	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mengubah informasi terkait dengan data yang ada.
11.	Menghapus Data Profil	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menghapus data yang sudah tidak ada.
12.	Mengisi Kuisioner	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mengisi kuisioner.
13.	Melihat Laporan Rekapitulasi Hasil Kuisioner	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, user eksternal, fakultas dan jurusan untuk melihat rekapitulasi hasil pengisian kuisioner.
14.	Mencetak Laporan Hasil	Sistem harus menyediakan fasilitas bagi

	Rekapitulasi Kuisioner	administrator, Alumni dan Mahasiswa Lulus, Fakultas dan Jurusan untuk mencetak laporan hasil rekapitulasi kuisioner sebagai arsip.
15.	Mengisi Forum Diskusi	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mengisi forum diskusi.
16.	Melihat Forum Diskusi	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, serta administrator untuk melihat forum diskusi.
17.	Menambah *Comment* Profil	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menambahkan comment kepada profil anggota lain.
18.	Menghapus *Comment* Profil	Sistem harus menyediakan fasilitas bagi administrator untuk menghapus comment yang tidak sesuai di profil anggota.
19.	Mencari Profil Alumni	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, Fakultas dan Jurusan, serta user eksternal untuk melakukan pencarian profil alumni tertentu.
20.	Melihat Profil Alumni	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, Fakultas dan Jurusan, serta *user* eksternal untuk melihat profil alumni.
21.	Mencetak Data Hasil Pencarian	Sistem harus menyediakan fasilitas bagi Alumni dan Mahasiswa Lulus, Fakultas dan Jurusan, serta user eksternal untuk mencetak data hasil pencarian.

22.	Menambah Data Fakultas	Sistem harus menyediakan fasilitas bagi Fakultas dan Jurusan untuk menambahkan data alumni masing-masing fakultas.
23.	Mengganti Data Fakultas	Sistem harus menyediakan fasilitas bagi Fakultas dan Jurusan untuk mengganti data alumni masing-masing fakultas.
24.	Menghapus Data Fakultas	Sistem harus menyediakan fasilitas bagi Fakultas dan Jurusan untuk menghapus data alumni masing-masing fakultas.
25.	Melihat Data Fakultas	Sistem harus menyediakan fasilitas bagi Fakultas dan Jurusan, Alumni dan Mahasiswa Lulus, dan user eksternal untuk menlihat data alumni masing-masing fakultas.

Sumber : [Analisis]

4.2 *Standard Operational Procedure* (SOP)

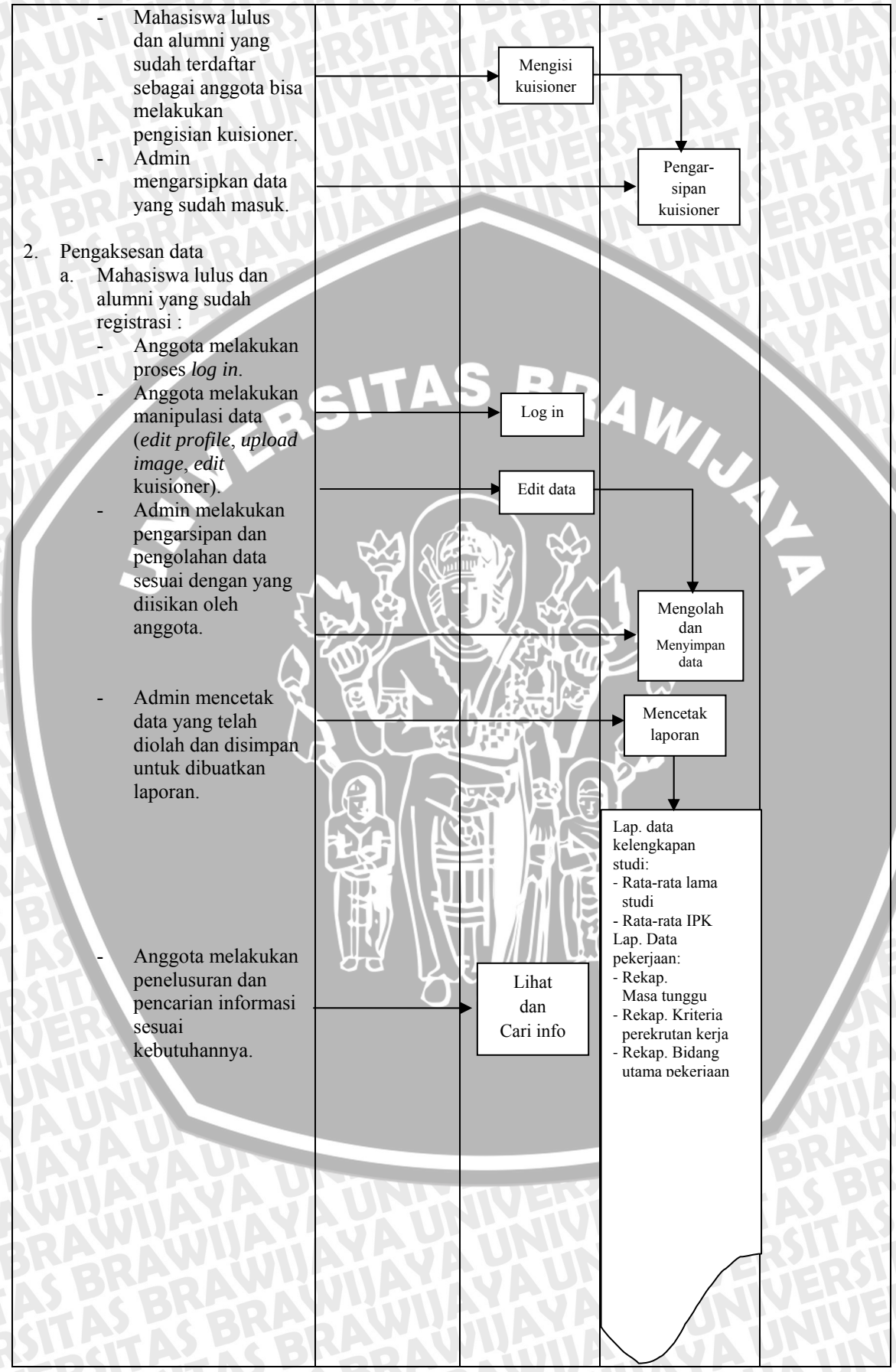
Perancangan dan pembuatan perangkat lunak yang akan dilakukan meliputi perancangan sistem yang dimulai dengan pembentukan SOP (*Standard Operational Procedure*). SOP adalah penetapan tertulis mengenai apa yang harus dilakukan, kapan, dimana, dan oleh siapa. SOP *Tracer Study* ditunjukkan pada Tabel 4.3 berikut:

Tabel 4.3 *Standard Operation Procedure (SOP) Elektronik Tracer Study*

Proses	User Eksternal	Alumni Dan Mahasiswa Lulus	Administrator (Puskom)	Fakultas/ Jurusan
1. Pengisian data kusisioner a. Melalui *web* - Mahasiswa lulus dan alumni melakukan registrasi sebagai anggota. - Admin melakukan validasi anggota.		Registrasi anggota	Validasi anggota	

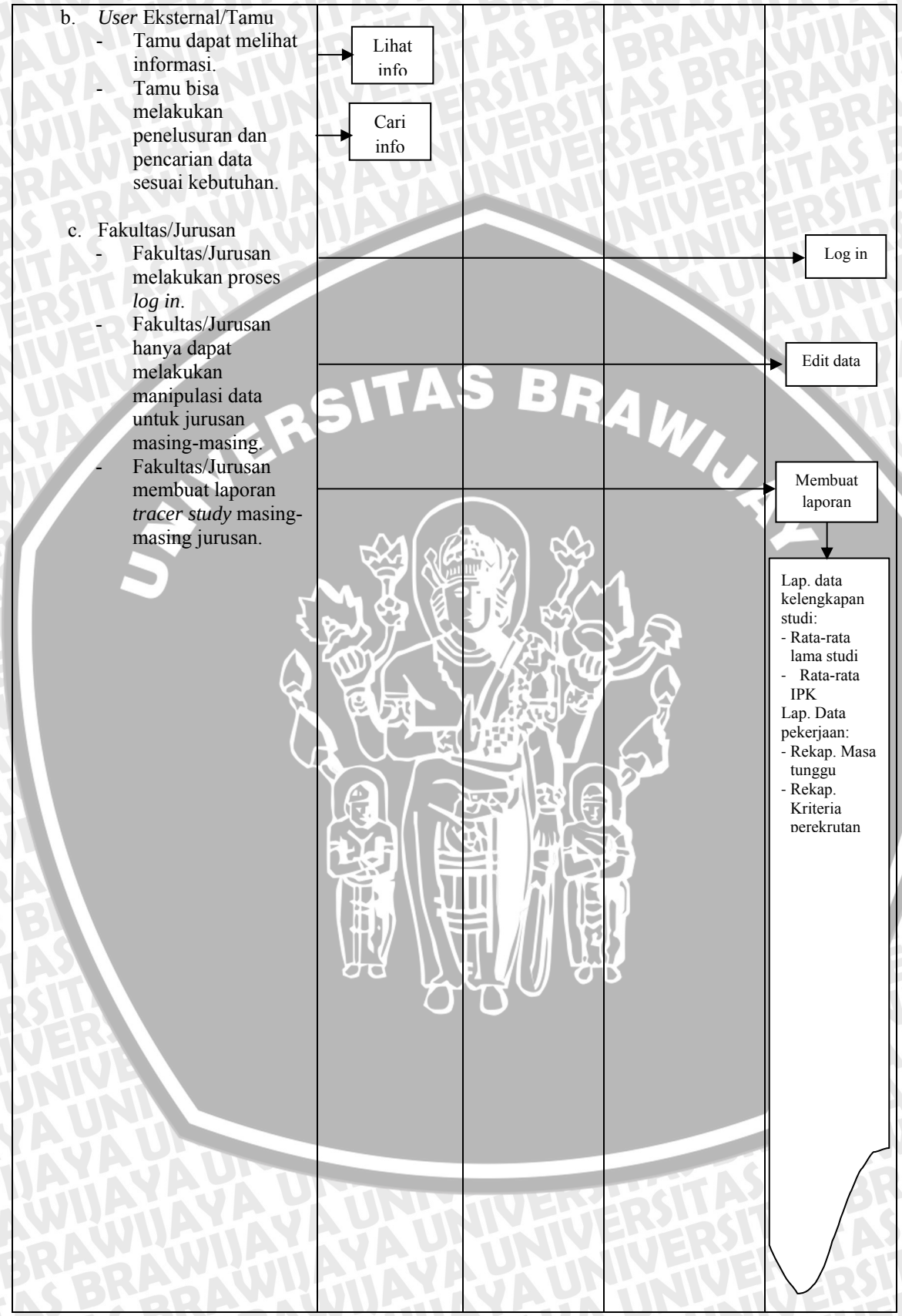
- Mahasiswa lulus dan alumni yang sudah terdaftar sebagai anggota bisa melakukan pengisian kuisioner.
- Admin mengarsipkan data yang sudah masuk.

2. Pengaksesan data
   a. Mahasiswa lulus dan alumni yang sudah registrasi :
      - Anggota melakukan proses *log in*.
      - Anggota melakukan manipulasi data (*edit profile*, *upload image*, *edit* kuisioner).
      - Admin melakukan pengarsipan dan pengolahan data sesuai dengan yang diisikan oleh anggota.
      - Admin mencetak data yang telah diolah dan disimpan untuk dibuatkan laporan.
      - Anggota melakukan penelusuran dan pencarian informasi sesuai kebutuhannya.



b. *User* Eksternal/Tamu
   - Tamu dapat melihat informasi.
   - Tamu bisa melakukan penelusuran dan pencarian data sesuai kebutuhan.

c. Fakultas/Jurusan
   - Fakultas/Jurusan melakukan proses *log in*.
   - Fakultas/Jurusan hanya dapat melakukan manipulasi data untuk jurusan masing-masing.
   - Fakultas/Jurusan membuat laporan *tracer study* masing-masing jurusan.



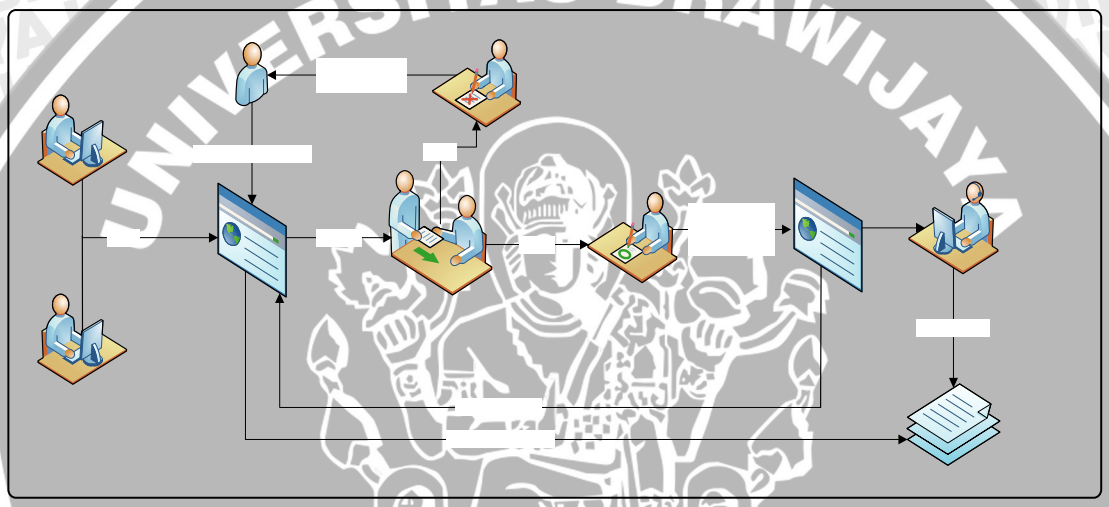
Sumber: [Analisis]

4.3 Perancangan Sistem

Perancangan ini dilakukan untuk mengetahui aplikasi sistem yang akan dibuat secara umum. Perancangan sistem meliputi diagram blok sistem, diagram konteks dan cara kerja sistem.

4.3.1 Blok Diagram Sistem

Dalam perancangan sistem disertakan Blok diagram sistem dari Sistem yang akan dibuat. Blok diagram aplikasi *tracer study* dapat ditunjukkan pada Gambar 4.2 berikut.



Gambar 4.2 *Blok Diagram Sistem*
Sumber: [Analisis]

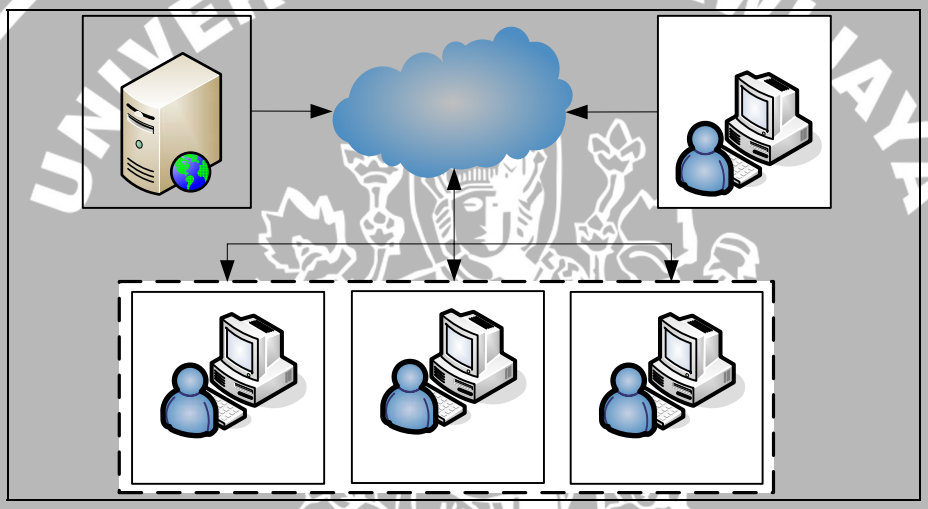
Diagram blok sistem aplikasi *tracer study* diawali dengan Alumni dan Mahasiswa Lulus yang harus melakukan registrasi anggota terlebih dahulu. *Account* yang telah divalidasi oleh administrator selanjutnya digunakan untuk mengakses *web tracer study*. Setelah *login*, Alumni dan Mahasiswa Lulus yang telah terdaftar sebagai anggota diharuskan mengisi kuisioner *tracer study*. Hasil pengisian kuisioner ini diolah oleh sistem secara *online* untuk kemudian ditampilkan hasilnya. Puskom sebagai administrator selanjutnya mencetak hasil rekapitulasi sebagai arsip.

Selain Alumni dan Mahasiswa Lulus, terdapat *user* lain yang dapat melakukan pengaksesan secara *online*. Fakultas dan Jurusan memiliki hak akses yang lebih tinggi dibandingkan *user* Alumni dan Mahasiswa Lulus. Fakultas dan Jurusan memiliki *account* tertentu yang sudah divalidasi oleh admin, sehingga dapat melakukan

manipulasi data. Hal tersebut berbeda dengan user eksternal yang hanya dapat mengakses data yang bersifat umum saja.

4.3.2 Arsitektur Jaringan Sistem

Aplikasi tracer study dibuat dengan menggunakan teknologi *web* yang akan dijalankan secara *online*. Sistem ini terdiri dari *web server* Apache, bahasa pemrograman PHP, basisdata MySQL dan Operating Sistem Windows XP yang diletakkan pada satu komputer yang berfungsi sebagai *server*. *Client* akan mengakses sistem informasi ini dengan menggunakan *browser*. Arsitektur sistem dapat digambarkan seperti yang diperlihatkan dalam Gambar 4.3 berikut.



Gambar 4.3 Arsitektur Jaringan Sistem

Sumber: [Analisis]

Arsitektur jaringan sistem aplikasi ini terdiri dari empat blok yang terhubung melalui jaringan Internet. Masing-masing blok dapat dijelaskan sebagai berikut:

1. Aplikasi Admin

   Aplikasi admin digunakan untuk mengelola data pribadi dan kuisioner yang telah diisikan *user*. Admin dapat menambahkan, menghapus, mengubah, ataupun melihat *account*. Selain itu, admin juga dapat melakukan penghapusan *comment* yang dianggap tidak layak untuk ditampilkan.

2. Aplikasi Alumni dan Mahasiswa Lulus

   Aplikasi alumni dan mahasiswa lulus digunakan oleh alumni dan mahasiswa lulus yang telah melakukan registrasi sebagai anggota untuk mengakses sistem. Sebagai anggota, alumni dan mahasiswa lulus dapat melakukan pengisian kuisioner, manipulasi data pribadi alumni, searching profil alumni lain, ataupun melakukan komunikasi dengan anggota lain melalui forum diskusi dan pertukaran *comment*.

3. Aplikasi Fakultas dan Jurusan

   Aplikasi fakultas/jurusan digunakan oleh fakultas dan jurusan untuk mengisikan data fakultas dan jurusan masing-masing ataupun alumni dari masing-masing fakultas. Fakultas dan Jurusan hanya dapat melakukan manipulasi data dari fakultas/jurusannya sendiri. Selain itu, fakultas dan jurusan juga dapat membuat laporan dari hasil rekapitulasi kuisioner yang dicetak sebagai arsip.

4. Aplikasi *User* Eksternal

   Aplikasi *user* eksternal digunakan oleh *user* yang tidak memiliki *account* di sistem ini. *User* eksternal hanya dapat melakukan pencarian data, melihat profil, dan mencetak hasil pencarian.

4.4 Perancangan Proses

Berdasarkan data dan informasi sebelumnya, perancangan sistem Aplikasi Web Penelusuran Data Alumni UB (Tracer Study) meliputi:

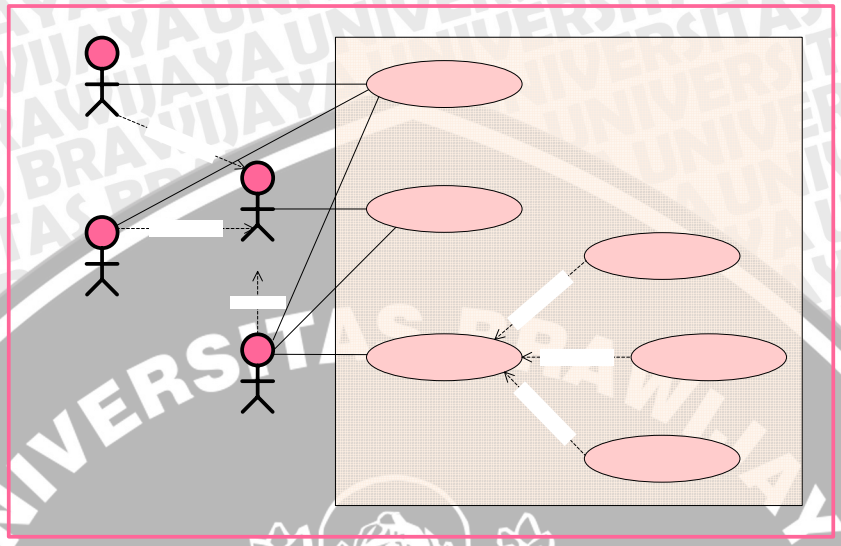
1. *Use Case Diagram*
2. *Class Diagram*
3. *Sequence Diagram*

4.4.1 *Use Case* Diagram

Pemodelan dalam use case diagram yang menggambarkan fungsionalitas yang disediakan oleh Aplikasi *Web Tracer Study* di bagi menjadi tiga buah diagram yang bersesuaian dengan kelompok kebutuhan.

4.4.1.1 *Use Case Diagram* untuk Administrasi Sistem *Tracer Study*

Diagram *use case* ini melibatkan tiga aktor yaitu Alumni dan Mahasiswa Lulus, Fakultas dan Jurusan, dan Administrator.



Gambar 4.4 *Use Case Diagram* untuk Administrasi Sistem *Tracer Study*

Sumber: [Analisis]

Berikut adalah spesifikasi use case untuk kegiatan administrasi sistem:

- Spesifikasi *Use Case Login*

Nama Use Case	Login
Aktor	User
Deskripsi	Sistem harus memberikan fasilitas untuk login, sehingga masing-masing aktor memiliki hak kerja masing-masing sesuai dengan peran.
Kondisi Awal	Aktor menjalankan sistem sampai pada halaman pertama
Kondisi Akhir	Aktor memasuki sistem dengan hak – hak tertentu sesuai kewenangannya.

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memasukkan username dan password, kemudian tekan tombol *"login"*	2. Sistem memvalidasi pasangan user name dan password yang dimasukan. 3. Sistem Menampilkan halaman, sesuai peran dari masing-masing aktor.
Pengecualian: Pasangan username dan password tidak ada dalam sistem	
	a.Sistem menampilkan pesan bahwa pasangan username dan password yang dimasukan tidak sesuai
	b.Sistem kembali ke langkah no 1

- Spesifikasi *Use Case Logout*

Nama Use Case	Logout
Aktor	User
Deskripsi	Sistem harus menyediakan fasilitas agar User dapat keluar dari sistem.
Kondisi Awal	User telah berada dalam sistem
Kondisi Akhir	User keluar dari sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1.Aktor menekan tombol *"Logout"*	2.Sistem melakukan Logout sehingga Aktor keluar dari sistem, dan sistem menampilkan Halaman Log in.

- Spesifikasi *Use Case* Melihat *Account*

Nama Use Case	Melihat *Account*
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas bagi administrator untuk melihat *account-account* yang ada.
Kondisi Awal	Aktor telah Login sebagai User yang berperan sebagai Administrator
Kondisi Akhir	Seluruh data *account* ditampilkan oleh sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1.Aktor memilih menu *"Lihat Account"*	2.Sistem menampilkan seluruh informasi *account* yang tersimpan dalam basis data.

- Spesifikasi *Use Case* Menambah *Account*

Nama Use Case	Menambahkan *Account*
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas bagi administrator untuk menambah *account* baru yang dapat melakukan akses terhadap sistem ini dengan hak-hak tertentu sesuai kapasitas dan keperluannya terhadap sistem ini.
Kondisi Awal	Aktor telah Login sebagai User yang berperan sebagai Administrator
Kondisi Akhir	Terdapat *account* baru dalam sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1.Aktor memilih menu *"Tambah Account"*	2.Sistem menampilkan form yang berfungsi untuk memasukan

	informasi nama, username, password, dan divisi.
3.Aktor memasukan informasi nama, username, password, dan divisi. Kemudian actor menekan tombol "Simpan"	4.Sistem menyimpan data (nama, username, password, dan divisi) dalam basis data. 5. Sistem Menampilkan pesan *"Tambah Data Berhasil"*.
Pengecualian: Username telah ada dalam sistem	
	a.Sistem tidak menyimpan informasi ke dalam sistem
	b.Sistem menampilkan pesan bahwa Username telah ada dalam sistem
	c. Sistem kembali menuju langkah no2.
Aturan Khusus - Tidak boleh ada kolom isian yang tidak terisi	

- Spesifikasi *Use Case* Mengganti *Account*

Nama Use Case	Mengubah *Account*
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas bagi administrator untuk mengubah *account-account* yang ada
Kondisi Awal	Aktor telah menjalankan *use case* Melihat *Account*
Kondisi Akhir	Informasi *account* yang diubah tersimpan
Aksi dari Aktor	**Tanggapan dari Sistem**
1.Aktor memilih salah satu baris *account*	2.Sistem menyajikan form berisi informasi *account* tersebut.
3.Aktor melakukan perubahan pada tempat yang tersedia, kemudian tekan tombol *"simpan"*.	4.Sistem menyimpan informasi form ke dalam basis data.
	5.Sistem menjalankan *use case* Melihat *Account*
Pengecualian: Perubahan menyebabkan adanya username ganda dalam system	
	a.Sistem tidak menyimpan perubahan yang dilakukan
	b.Sistem menampilkan pesan bahwa username tersebut telah ada dalam system

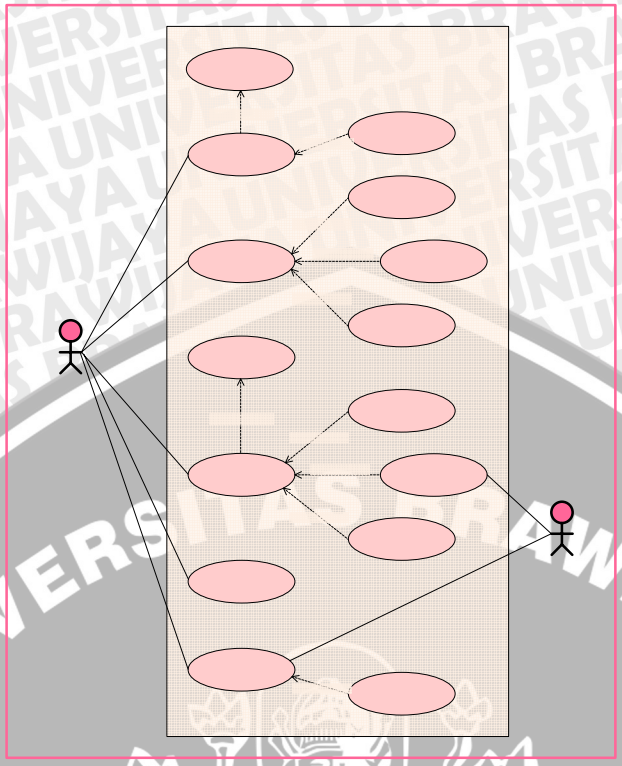
	c.Sistem menjalankan *use case* Melihat *Account*
Aturan Khusus - Tidak boleh ada kolom isian yang tidak terisi	

- Spesifikasi *Use Case* Menghapus *Account*

Nama Use Case	Menghapus *Account*
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas bagi administrator untuk menghapus *account-account* yang ada
Kondisi Awal	Aktor telah menjalankan *use case* Melihat *Account*
Kondisi Akhir	*Account* yang dipilih terhapus dari basis data sistem.
Aksi dari Aktor	**Tanggapan dari Sistem**
1.Aktor memilih salah satu baris *account,* kemudian tekan pilihan*"hapus"*.	2.Sistem menghapus data *account* telah dipilih.
	3.Sistem menjalankan *use case* Melihat *Account*

4.4.1.2 *Use Case Diagram* untuk Anggota *Tracer Study*

Use Case Diagram untuk kebutuhan fungsional yang termasuk dalam pendaftaran anggota sistem aplikasi *web Tracer Study* melibatkan dua aktor yaitu Alumni dan Mahasiswa Lulus, dan Administrator.



Gambar 4.5 *Use Case Diagram* Anggota *Tracer Study*

Sumber: [Analisis]

Berikut adalah spesifikasi *use case* untuk kegiatan pendaftaran anggota *tracer study*:

- Spesifikasi *Use Case* Mengisi Data Registrasi

Nama Use Case	Mengisi Data Registrasi
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas untuk melakukan registrasi sebagai anggota tracer study.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Alumni dan Mahasiswa Lulus
Kondisi Akhir	Aktor telah mengisi data registrasi

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor menekan tombol ”Daftar”	2. Sistem menampilkan form registrasi
3. Aktor memasukkan data kelengkapan studi, kemudian menekan tombol ”Simpan”.	4. Sistem menyimpan data ke dalam basis data. 5. Sistem menampilkan pesan bahwa data berhasil disimpan.
Pengecualian: Data tidak diisi seluruhnya	

	a.Sistem menampilkan pesan bahwa ada kolom yang belum terisi.
	b.Sistem menampilkan kolom yang belum terisi.

- Spesifikasi *Use Case* Mengisi Kuisioner

Nama Use Case	Mengisi Kuisioner
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melakukan proses pengisian kuisioner.
Kondisi Awal	Aktor telah mengisi data registrasi
Kondisi Akhir	Kuisioner direkapitulasi dalam bentuk tabel
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor mengisi kuisioner, kemudian menekan tombol ”Submit”	2. Sistem menyimpan data dalam basis data dan mengolah data hasil kuisioner. 3. Sistem menampilkan pesan bahwa data berhasil disimpan. 4. Sistem menampilkan halaman hasil rekapitulasi kuisioner.
Pengecualian: Pertanyaan kuisioner tidak diisi seluruhnya	
	a.Sistem menampilkan pesan bahwa ada pertanyaan yang belum terjawab.
	b.Sistem menampilkan halaman awal.

- Spesifikasi *Use Case* Melihat Hasil Kuisioner

Nama Use Case	Melihat Hasil Kuisioner
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melihat laporan rekapitulasi hasil kuisioner.
Kondisi Awal	Aktor telah login sebagai Alumni dan Mahasiswa Lulus
Kondisi Akhir	Hasil rekapitulasi kuisioner ditampilkan dalam bentuk tabel
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu ”Laporan”	2. Sistem menampilkan halaman hasil rekapitulasi kuisioner.

- Spesifikasi *Use Case* Melihat Data Profil

Nama Use Case	Melihat Data Profil
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melihat data profil miliknya.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Alumni dan Mahasiswa Lulus.
Kondisi Akhir	Sistem menampilkan data profil milik aktor
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu Profil, kemudian menekan tombol ”Lihat Profil”	2. Sistem menampilkan data profil aktor seluruhnya.

- Spesifikasi *Use Case* Menambah Data Profil

Nama Use Case	Menambah Data Profil
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menambah data profil.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Alumni dan Mahasiswa Lulus
Kondisi Akhir	Terdapat data profil baru dalam basis data sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu Profil, kemudian tekan tombol ”Tambah Profil”	2. Sistem menampilkan form data profil alumni yang berisi data pribadi dan data pekerjaan.
3. Aktor memasukkan data pribadi dan data pekerjaan, kemudian tekan tombol ”Simpan”	4. Sistem menyimpan data dalam basis data. 5. Sistem menampilkan pesan ”Penambahan Data Berhasil”, dan sistem menjalankan use case Melihat Profil Alumni.
Aturan Khusus: Tidak boleh ada kolom yang tidak terisi	

- Spesifikasi *Use Case* Mengganti Data Profil

Nama Use Case	Mengganti Data Profil
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mengganti data profil.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Data Profil”
Kondisi Akhir	Data yang telah diganti disimpan ke dalam sistem

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih salah satu baris pada baris data profil, kemudian menekan tombol ”Edit”.	2. Sistem menyajikan informasi tersebut dalam bentuk form.
3. Aktor melakukan perubahan pada form yang tersedia, kemudian tekan tombol ”Simpan”	4. Sistem menyimpan data dalam basis data.
	5. Sistem menampilkan pesan bahwa data berhasil diubah, dan sistem menjalankan use case ”Melihat Data Profil”.
Aturan Khusus: Tidak boleh ada kolom yang tidak terisi	

- Spesifikasi *Use Case* Menghapus Data Profil

Nama Use Case	Menghapus Data Profil
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menghapusdata profil.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Data Profil”
Kondisi Akhir	Data yang dipilih terhapus dari basis data sistem

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih salah satu baris pada baris data profil, kemudian menekan tombol ”Hapus”.	2. Sistem menampilkan pesan konfirmasi penghapusan data.
3. Aktor memilih tombol ”Yes”.	4. Sistem menampilkan pesan konfirmasi bahwa data telah dihapus. 5. Data yang dipilih dihapus dari sistem.
	6. Sistem menjalankan *use case* ”Melihat Data Profil”.

- Spesifikasi *Use Case* Mencari Profil Alumni

Nama Use Case	Mencari Profil Alumni
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melakukan pencarian profil alumni lain.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Alumni dan Mahasiswa Lulus
Kondisi Akhir	Profil alumni yang dicari ditampilkan

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu pencarian berdasarkan Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun Lulus, atau Kota tempat tinggal.	2. Sistem menampilkan kategori pencarian data.
3. Aktor memasukkan data yang akan dicari berdasarkan kategori yang telah dipilih, kemudian tekan tombol ”Cari”.	4. Sistem menampilkan profil alumni yang berisi data Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun lulus, dan Kota Tempat Tinggal
Pengecualian I: Data yang dimasukkan tidak sesuai kategori	
	a. Sistem menampilkan pesan bahwa data yang dicari tidak ada. b. Sistem kembali menampilkan halaman pencarian.
Pengecualian II: Data yang dicari tidak terdapat dalam basis data sistem	
	a. Sistem menampilkan pesan bahwa data yang dicari tidak ada. b. Sistem kembali menampilkan halaman pencarian.

- Spesifikasi *Use Case* Melihat Profil Alumni

Nama Use Case	Melihat Profil Alumni
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melihat profil alumni lain
Kondisi Awal	Aktor telah menjalankan *use case* ”Mencari Profil Alumni”
Kondisi Akhir	Sistem menampilkan data profil alumni

Aksi dari Aktor	**Tanggapan dari Sistem**

1. Aktor memilih tombol ”Lihat Profil” pada hasil pencarian.	3. Sistem menampilkan data profil umum alumni.

- Spesifikasi *Use Case* Menambah *Comment*

Nama Use Case	Menambah *Comment*
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk menambah *comment* pada profil alumni lain.
Kondisi Awal	Aktor telah login sebagai Alumni dan Mahasiswa Lulus, dan aktor telah menjalankan *use case* ”Melihat Profil Alumni”
Kondisi Akhir	*Comment* yang ditambahkan ditampilkan dalam profil alumni oleh sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Comment”	2. Sistem menampilkan form *comment*.
3. Aktor mengisikan komentar pada form *comment*, kemudian tekan tombol ”Submit”.	4. Sistem menyimpan data dalam basis data. 5. Sistem menampilkan pesan bahwa *comment* berhasil dimasukkan. 6. Sistem menampilkan *comment* yang diisikan.
	7. Sistem menjalankan *use case* ”Melihat Profil Alumni”.
Pengecualian: Form comment tidak diisi	
	a. Sistem menampilkan pesan bahwa form comment belum diisi. b. Sistem menjalankan *use case* ”Melihat Profil Alumni”.

- Spesifikasi *Use Case* Menghapus *Comment*

Nama Use Case	Menghapus *Comment*
Aktor	Alumni dan Mahasiswa Lulus, Administrator
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus, Administrator untuk menghapus *comment* pada profil alumni.
Kondisi Awal	Aktor telah login sebagai Alumni dan Mahasiswa Lulus dan Administrator
Kondisi Akhir	*Comment* dihapus dari basis data sistem.
Aksi dari Aktor	**Tanggapan dari Sistem**

1. Aktor memilih salah satu baris pada baris comment profil, kemudian pilih tombol ”Hapus”	2. Sistem menampilkan pesan konfirmasi penghapusan *comment*.
3. Aktor memilih tombol ”Yes”.	4. Sistem menampilkan pesan konfirmasi bahwa *comment* telah dihapus. 5. Comment yang dipilih dihapus dari basis data sistem.
	6. Sistem menjalankan *use case* ”Melihat Profil Alumni”.

- Spesifikasi *Use Case* Mencetak Profil Alumni

Nama Use Case	Mencetak Profil Alumni
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mencetak halaman profil alumni.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Hasil Kuisioner”
Kondisi Akhir	Halaman profil alumni dapat tercetak
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Print”	2. Sistem akan menyajikan informasi dalam sebuah halaman dengan format seperti pada hasil pencetakan dan dilanjutkan dengan mencetak halaman tersebut.

- Spesifikasi *Use Case* Melihat Data Fakultas

Nama Use Case	Melihat Data Fakultas
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk melihat data Fakultas.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Alumni dan Mahasiswa Lulus.
Kondisi Akhir	Sistem menampilkan data Alumni Fakultas tertentu
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu Fakultas.	2. Sistem menampilkan data nama-nama Fakultas yang ada dalam basis data sistem.
3. Aktor memilih salah satu baris pada baris data fakultas.	4. Sistem menampilkan halaman daftar Alumni Fakultas yang dipilih.

- Spesifikasi *Use Case* Melihat Forum Diskusi

Nama Use Case	Melihat Forum Diskusi
Aktor	Alumni dan Mahasiswa Lulus, Administrator
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus dan Administrator untuk melihat forum diskusi.
Kondisi Awal	Aktor telah login sebagai Alumni dan Mahasiswa Lulus, dan Administrator
Kondisi Akhir	Sistem menampilkan forum diskusi secara keseluruhan

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu ”Forum Diskusi”.	2. Sistem menampilkan halaman forum diskusi.
3. Aktor memilih topik diskusi.	4. Sistem menampilkan halaman topik diskusi yang dipilih.

- Spesifikasi *Use Case* Mengisi Forum Diskusi

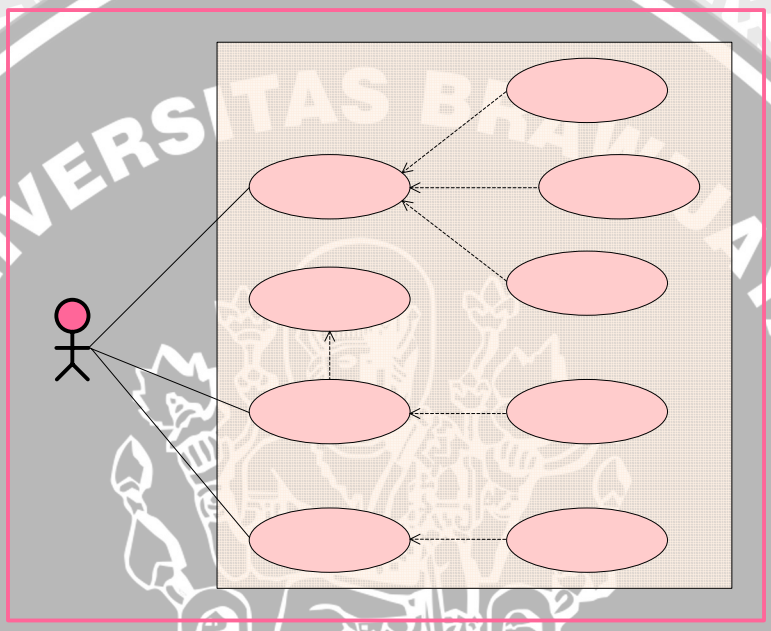
Nama Use Case	Mengisi Forum Diskusi
Aktor	Alumni dan Mahasiswa Lulus
Deskripsi	Sistem harus memberikan fasilitas bagi Alumni dan Mahasiswa Lulus untuk mengisi forum diskusi.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Forum Diskusi”
Kondisi Akhir	Data yang diisikan ditampilkan pada halaman forum diskusi oleh sistem

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih salah satu baris pada baris topik forum diskusi.	2. Sistem menampilkan halaman topik yang dipilih dan form diskusi
3. Aktor mengisi form diskusi, kemudian menekan tombol ”Submit”.	4. Sistem menyimpan data dalam basis data. 5. Sistem menampilkan pesan bahwa data berhasil dimasukkan. 6. Sistem menampilkan halaman topik diskusi.
	7. Sistem menjalankan *use case* ”Melihat Profil Alumni”.
Pengecualian: User tidak memiliki *account* sebagai Alumni dan Mahasiswa Lulus	
	c. Sistem menampilkan pesan bahwa user belum terdaftar. d. Sistem menjalankan *use case*

	”Melihat Forum Diskusi”.

4.4.1.3 *Use Case Diagram* untuk Kegiatan Pengelolaan Data Fakultas Sistem *Tracer Study*

Use Case Diagram untuk kebutuhan fungsional yang termasuk dalam kegiatan pengelolaan data Fakultas sistem *Tracer Study* hanya melibatkan satu aktor saja yaitu Fakultas dan Jurusan.



Gambar 4.6 *Use Case Diagram* untuk Kegiatan Pengelolaan Data Fakultas Sistem *Tracer Study*
Sumber: [Analisis]

Berikut adalah spesifikasi *use case* untuk kegiatan pendaftaran anggota *tracer study*:

- Spesifikasi *Use Case* Melihat Data Fakultas

Nama Use Case	Melihat Data Fakultas
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk melihat data Fakultas.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Fakultas

Fakultas
dan
Jurusan

	dan Jurusan
Kondisi Akhir	Sistem menampilkan data Alumni Fakultas tertentu

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu Fakultas.	2. Sistem menampilkan data nama-nama Fakultas yang ada dalam basis data sistem.
3. Aktor memilih salah satu baris pada baris data fakultas.	4. Sistem menampilkan halaman daftar Alumni Fakultas yang dipilih.

- Spesifikasi *Use Case* Menambah Data Fakultas

Nama Use Case	Menambah Data Fakultas
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk menambah data Fakultas
Kondisi Awal	Aktor telah login sebagai Fakultas dan Jurusan
Kondisi Akhir	Terdapat data Alumni Fakultas baru dalam basis data sistem

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Tambah Alumni”	2. Sistem menampilkan form data alumni yang terdiri dari Nama, NIM, Fakultas, Jurusan, Tahun Masuk,dan Tahun Lulus.
3. Aktor memasukkan alumni, kemudian tekan tombol ”Simpan”.	4. Sistem menyimpan data dalam basis data. 5. Sistem menampilkan pesan ”Penambahan Data Berhasil”. 6. Sistem menjalankan *use case* ”Melihat Data Fakultas”
Pengecualian: Aktor tidak terdaftar sebagai pengelola Fakultas	
	a. Sistem menampilkan pesan bahwa user tidak dapat melakukan penambahan data. b. Sistem menjalankan *use case* ”Melihat Data Fakultas”.
Aturan Khusus: Tidak boleh ada kolom yang tidak terisi	

- Spesifikasi *Use Case* Mengganti Data Fakultas

Nama Use Case	Mengganti Data Fakultas
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk mengganti data Fakultas.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Data Profil”
Kondisi Akhir	Data yang telah diganti disimpan ke dalam sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih salah satu baris pada baris data Fakultas.	2. Sistem menampilkan halaman daftar Alumni Fakultas yang dipilih.
3. Aktor memilih salah satu baris pada baris daftar Alumni Fakultas, kemudian menekan tombol ”Edit”.	4. Sistem menampilkan data tersebut dalam bentuk form.
5. Aktor melakukan perubahan pada form tersebut, kemudian menekan tombol ”Simpan”.	6. Sistem menyimpan data dalam basis data sistem. 7. Sistem menjalankan *use case* ”Melihat Data Fakultas”.
Aturan Khusus: Tidak boleh ada kolom yang tidak terisi	

- Spesifikasi *Use Case* Menghapus Data Fakultas

Nama Use Case	Menghapus Data Fakultas
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk menghapusdata Fakultas.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Data Fakultas”
Kondisi Akhir	Data yang dipilih terhapus dari basis data sistem
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih salah satu baris pada baris data Fakultas.	2. Sistem menampilkan halaman daftar Alumni Fakultas yang dipilih.
3. Aktor memilih salah satu baris pada baris daftar Alumni Fakultas, kemudian menekan tombol ”Hapus”.	4. Sistem menampilkan pesan konfirmasi penghapusan data.
5. Aktor memilih tombol ”Yes”.	6. Sistem menampilkan pesan konfirmasi bahwa data telah dihapus.

	7. Data yang dipilih dihapus dari sistem.
	8. Sistem menjalankan *use case* ”Melihat Data Fakultas”.

- Spesifikasi *Use Case* Mencari Profil Alumni

Nama Use Case	Mencari Profil Alumni
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk melakukan pencarian profil alumni.
Kondisi Awal	Aktor telah login sebagai user yang berperan sebagai Fakultas dan Jurusan
Kondisi Akhir	Profil alumni yang dicari ditampilkan
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu pencarian berdasarkan Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun Lulus, atau Kota tempat tinggal.	2. Sistem menampilkan kategori pencarian data.
3. Aktor memasukkan data yang akan dicari berdasarkan kategori yang telah dipilih, kemudian tekan tombol ”Cari”.	4. Sistem menampilkan profil alumni yang berisi data Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun lulus, dan Kota Tempat Tinggal
Pengecualian I: Data yang dimasukkan tidak sesuai kategori	
	a. Sistem menampilkan pesan bahwa data yang dicari tidak ada. b. Sistem kembali menampilkan halaman pencarian.
Pengecualian II: Data yang dicari tidak terdapat dalam basis data sistem	
	a. Sistem menampilkan pesan bahwa data yang dicari tidak ada. b. Sistem kembali menampilkan halaman pencarian.

- Spesifikasi *Use Case* Melihat Profil Alumni

Nama Use Case	Melihat Profil Alumni
Aktor	Fakultas dan Jurusan

Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk melihat profil alumni lain
Kondisi Awal	Aktor telah menjalankan *use case* ”Mencari Profil Alumni”
Kondisi Akhir	Sistem menampilkan data profil alumni
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Lihat Profil” pada hasil pencarian.	2. Sistem menampilkan data profil umum alumni.

- Spesifikasi *Use Case* Mencetak Profil Alumni

Nama Use Case	Mencetak Profil Alumni
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk mencetak halaman profil alumni.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Hasil Kuisioner”
Kondisi Akhir	Halaman profil alumni dapat tercetak
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Print”	2. Sistem akan menyajikan informasi dalam sebuah halaman dengan format seperti pada hasil pencetakan dan dilanjutkan dengan mencetak halaman tersebut.

- Spesifikasi *Use Case* Melihat Hasil Kuisioner

Nama Use Case	Melihat Hasil Kuisioner
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk melihat laporan rekapitulasi hasil kuisioner.
Kondisi Awal	Aktor telah login sebagai Fakultas dan Jurusan
Kondisi Akhir	Hasil rekapitulasi kuisioner ditampilkan dalam bentuk tabel
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu ”Laporan”	2. Sistem menampilkan halaman hasil rekapitulasi kuisioner.

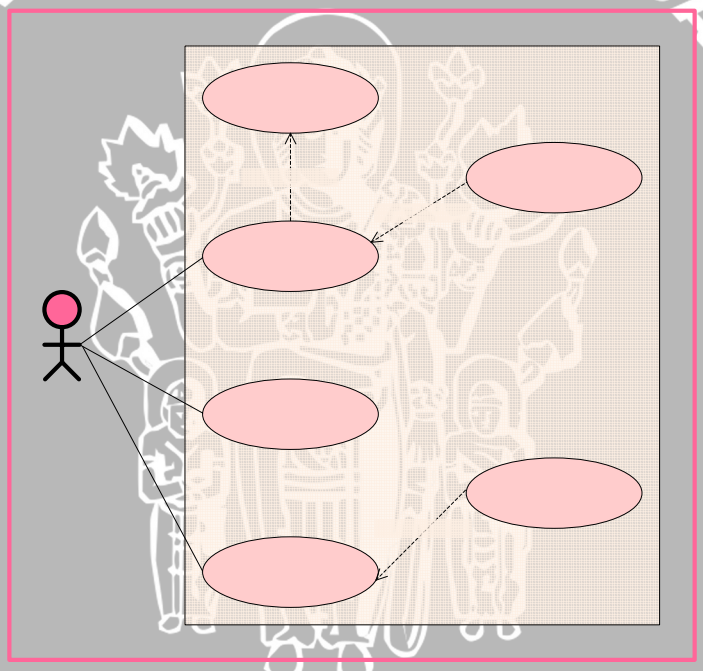
- Spesifikasi *Use Case* Mencetak Hasil Kuisioner

Nama Use Case	Mencetak Hasil Kuisioner
Aktor	Fakultas dan Jurusan
Deskripsi	Sistem harus memberikan fasilitas bagi Fakultas dan Jurusan untuk mencetak laporan rekapitulasi hasil kuisioner.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Hasil Kuisioner”

Kondisi Akhir	Hasil rekapitulasi kuisioner dapat tercetak
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Print”	2. Sistem akan menyajikan informasi dalam sebuah halaman dengan format seperti pada hasil pencetakan dan dilanjutkan dengan mencetak halaman tersebut.

4.4.1.4 *Use Case Diagram* Pengunjung Sistem Aplikasi *Web Tracer Study*

Use Case Diagram untuk kebutuhan fungsional yang termasuk dalam pengunjung sistem aplikasi *web Tracer Study* hanya melibatkan satu aktor saja yaitu User Eksternal.



Gambar 4.7 *Use Case Diagram* Pengunjung Sistem Aplikasi *Web Tracer Study*
Sumber: [Analisis]

Berikut adalah spesifikasi *use case* untuk pengunjung sistem aplikasi *web tracer study*:

- Spesifikasi *Use Case* Mencari Profil Alumni

Nama Use Case	Mencari Profil Alumni
Aktor	User Eksternal
Deskripsi	Sistem harus memberikan fasilitas bagi User Eksternal untuk melakukan pencarian profil alumni.

Kondisi Awal	Aktor menjalankan sistem sampai halaman pertama
Kondisi Akhir	Profil alumni yang dicari ditampilkan
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu pencarian berdasarkan Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun Lulus, atau Kota tempat tinggal.	2. Sistem menampilkan kategori pencarian data.
3. Aktor memasukkan data yang akan dicari berdasarkan kategori yang telah dipilih, kemudian tekan tombol ”Cari”.	4. Sistem menampilkan profil alumni yang berisi data Nama, NIM, Fakultas, Jurusan, Tahun Masuk, Tahun lulus, dan Kota Tempat Tinggal
Pengecualian I: Data yang dimasukkan tidak sesuai kategori	
	c. Sistem menampilkan pesan bahwa data yang dicari tidak ada. d. Sistem kembali menampilkan halaman pencarian.
Pengecualian II: Data yang dicari tidak terdapat dalam basis data sistem	
	c. Sistem menampilkan pesan bahwa data yang dicari tidak ada. d. Sistem kembali menampilkan halaman pencarian.

- Spesifikasi *Use Case* Melihat Profil Alumni

Nama Use Case	Melihat Profil Alumni
Aktor	User Eksternal
Deskripsi	Sistem harus memberikan fasilitas bagi User Eksternal untuk melihat profil alumni lain
Kondisi Awal	Aktor telah menjalankan *use case* ”Mencari Profil Alumni”
Kondisi Akhir	Sistem menampilkan data profil alumni
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Lihat Profil” pada hasil pencarian.	2. Sistem menampilkan data profil umum alumni.

- Spesifikasi *Use Case* Mencetak Profil Alumni

Nama Use Case	Mencetak Profil Alumni
Aktor	User Eksternal

Deskripsi	Sistem harus memberikan fasilitas bagi User Eksternal untuk mencetak halaman profil alumni.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Hasil Kuisioner”
Kondisi Akhir	Halaman profil alumni dapat tercetak

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Print”	2. Sistem akan menyajikan informasi dalam sebuah halaman dengan format seperti pada hasil pencetakan dan dilanjutkan dengan mencetak halaman tersebut.

- Spesifikasi *Use Case* Melihat Hasil Kuisioner

Nama Use Case	Melihat Hasil Kuisioner
Aktor	User Eksternal
Deskripsi	Sistem harus memberikan fasilitas bagi User Eksternal untuk melihat laporan rekapitulasi hasil kuisioner.
Kondisi Awal	Aktor menjalankan sistem sampai halaman pertama
Kondisi Akhir	Hasil rekapitulasi kuisioner ditampilkan dalam bentuk tabel

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih menu ”Laporan”	2. Sistem menampilkan halaman hasil rekapitulasi kuisioner.

- Spesifikasi *Use Case* Mencetak Hasil Kuisioner

Nama Use Case	Mencetak Hasil Kuisioner
Aktor	User Eksternal
Deskripsi	Sistem harus memberikan fasilitas bagi User Eksternal untuk mencetak laporan rekapitulasi hasil kuisioner.
Kondisi Awal	Aktor telah menjalankan *use case* ”Melihat Hasil Kuisioner”
Kondisi Akhir	Hasil rekapitulasi kuisioner dapat tercetak

Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memilih tombol ”Print”	2. Sistem akan menyajikan informasi dalam sebuah halaman dengan format seperti pada hasil pencetakan dan dilanjutkan dengan mencetak halaman tersebut.

4.4.2 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) [AMR-02].

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Kandidat klas diperoleh melalui kosa kata yang merupakan jenis kata benda yang dapat kita temukan dalam penjelasan sistem yang akan kita bangun atau dapat juga melalui *use case* yang telah dibuat sebelumnya. Setiap kata benda yang ditemukan dapat dikenali karakteristiknya, karakteristik ini yang menjadi atribut bagi klas yang dirancang. Masing-masing kata benda yang telah ditemukan tentunya memiliki peran yang dapat dikenali dalam kaitannya dengan sistem yang akan dibangun, peran inilah yang nantinya akan menjadi operasi atau *method* dalam klas yang dirancang.

Pada tabel dibawah ini yaitu tabel 4.4 akan ditampilkan *use case-use case* yang ada dalam sistem yang akan dibangun. Dari daftar *use case* tersebut akan didapatkan kata benda-kata benda yang menjadi kandidat klas.

Tabel 4.4 Daftar *Use Case* dan Kandidat *Class*

Use Case	**Kandidat *Class***
1. Login	▪ User
2. Logout	▪ User
3. Melihat *Account*	▪ Administrator
	▪ Account
4. Menambah *Account*	▪ Administrator
	▪ Account

5. Mengganti *Account*	▪ Administrator ▪ Account
6. Menghapus *Account*	▪ Administrator ▪ Account
7. Mengisi Data Registrasi	▪ Alumni ▪ Registrasi
8. Mengisi Kuisioner	▪ Alumni ▪ Kuisioner
9. Melihat Hasil Kuisioner	▪ Alumni ▪ Fakultas ▪ User Eksternal ▪ Kuisioner
10. Mencetak Hasil Kuisioner	▪ Fakultas ▪ User Eksternal ▪ Kuisioner ▪ Cetak
11. Melihat Data Profil	▪ Alumni ▪ Profil
12. Menambah Data Profil	▪ Alumni ▪ Profil
13. Mengganti Data Profil	▪ Alumni ▪ Profil
14. Menghapus Data Profil	▪ Alumni ▪ Profil
15. Mencari Profil Alumni	▪ Alumni ▪ Fakultas ▪ User Eksternal ▪ Profil
16. Melihat Profil Alumni	▪ Alumni ▪ Fakultas ▪ User Eksternal ▪ Profil
17. Menambah Comment Profil	▪ Alumni ▪ Profil ▪ Comment
18. Menghapus Comment Profil	▪ Alumni ▪ Administrator ▪ Profil ▪ Comment
19. Mencetak Profil Alumni	▪ Alumni ▪ Fakultas ▪ User Eksternal ▪ Profil ▪ Cetak
20. Melihat Data Fakultas	▪ Alumni ▪ Fakultas

	▪ User Eksternal
	▪ Daftar alumni
21. Menambah Data Fakultas	▪ Fakultas
	▪ Daftar alumni
22. Mengganti Data Fakultas	▪ Fakultas
	▪ Daftar alumni
23. Menghapus Data Fakultas	▪ Fakultas
	▪ Daftar alumni
24. Melihat Forum Diskusi	▪ Alumni
	▪ Administrator
	▪ Fordis
25. Mengisi Forum Diskusi	▪ Alumni
	▪ Fordis

Sumber: [Analisis]

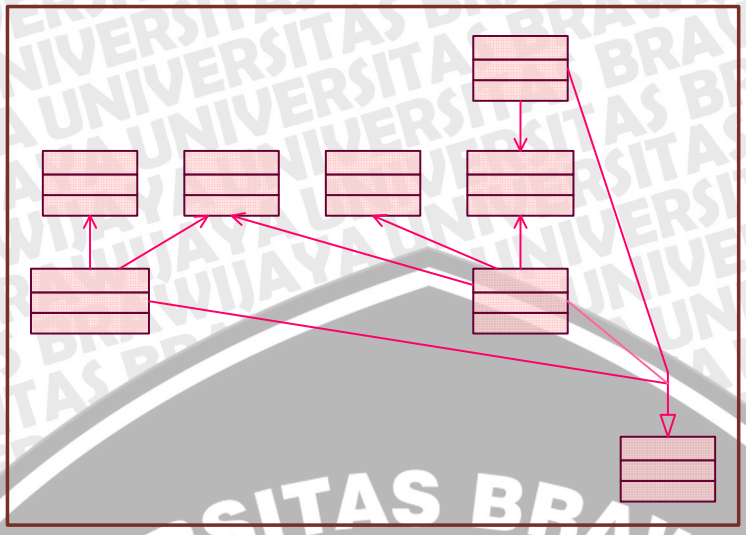
Dari tabel diatas yang memuat kandidat-kandidat klas bagi sistem yang akan dibangun, maka dapat diperoleh daftar klas yang diperlukan untuk sistem yang akan dibangun. Klas-klas tersebut dapat ditampilkan pada daftar berikut ini.

Tabel 4.5 Daftar *Class* yang Dibutuhkan oleh Sistem

No.	*Class* yang Dibutuhkan Sistem
1.	Account
2.	Alumni
3.	Kuisioner
4.	Comment
5.	Fakultas
6.	Fordis

Sumber: [Analisis]

Relasi antar klas tersebut dapat dilihat pada diagram klas dibawah ini. Atribut dan operasi masing-masing klas akan disampaikan pada penjelasan masing-masing klas, hal ini dilakukan agar dapat menampilkan gambar secara sederhana yang mudah dimengerti.



Gambar 4.8 *Class Diagram*
Sumber: [Analisis]

4.4.2.1 Penjelasan *Class Diagram* untuk Klas *Account*

Klas ini berguna untuk memfasilitasi pencatatan *user* yang diizinkan untuk menggunakan sistem kedalam basis data.

Account
- *private intIdAlumni :int* - *private intIdFakultas :int* - *private strUsername :Str* - *private strPassword :Str*
+ *memasukkan_account ( ) : String* + *mengubah _account ( ) : String* + *menghapus_ account ( ) : String* + *melihat_ account ( ) : String*

4.4.2.2 Penjelasan *Class Diagram* untuk Klas *Comment*

Klas ini berguna untuk mengolah *comment* yang diisikan dalam profil alumni.

Comment
- *private intIdComment :int*

- *private sttData :str*
+ *menambah_comment ( ) : String* + *menghapus_ comment ( ) : String*

4.4.2.3 Penjelasan *Class Diagram* untuk Klas Fordis

Klas ini digunakan untuk mengolah data yang diisikan dalam Forum Diskusi.

Fordis
- *private intIdFordis :int* - *private sttData :str*
+ *melihat_fordis ( ) : String* + *mengisi_fordis ( ) : String*

4.4.2.4 Penjelasan *Class Diagram* untuk Klas Kuisioner

Klas ini digunakan untuk mengolah data kusioner dan melihat hasil pengolahan data kuisioner.

Kuisioner
- *private intIdKuisioner :int* - *private sttData :str*
+ *mengisi_kuisioner ( ) : String* + *melihat_hasilKuisioner ( ) : String* + *mencetak_hasilKuisioner ( ) : String*

4.4.2.5 Penjelasan *Class Diagram* untuk Klas Alumni

Klas ini berfungsi untuk menginstansi Klas Comment dan Klas Fordis yang akan memanfaatkan operasi-operasi yang dimilikinya.

Alumni
- *private intIdAlumni :int*

- *private sttData :str*
+ *mengisi_registrasi ( ) : String* + *mengisi_kuisioner ( ) : String* + *melihat_kuisioner ( ) : String* + *melihat_profil ( ) : String* + *menambah_profil ( ) : String* + *mengganti_profil ( ) : String* + *menghapus_profil ( ) : String* + *mencari_alumni ( ) : String* + *melihat_alumni ( ) : String* + *menambah_comment ( ) : String* + *menghapus_ comment ( ) : String* + *mencetak_profil ( ) : String* + *melihat_dataFakultas ( ) : String* + *melihat_fordis ( ) : String* + *mengisi_fordis ( ) : String*

4.4.2.6 Penjelasan *Class Diagram* untuk Klas Fakultas

Klas ini berfungsi untuk menginstansi klas Alumni dan akan memanfaatkan operasi-operasi yang dimilikinya.

Fakultas
- *private intIdFakultas :int* - *private sttData :str*
+ *melihat_hasilKuisioner ( ) : String* + *mencetak_hasilKuisioner ( ) : String* + *mencari_alumni ( ) : String* + *melihat_alumni ( ) : String* + *mencetak_profil ( ) : String* + *melihat_dataFakultas ( ) : String* + *menambah_dataFakultas ( ) : String* + *mengganti_dataFakultas ( ) : String*

+ *menghapus_ dataFakultas ( ) : String*

4.4.2.7 Penjelasan *Class Diagram* untuk Klas *User*

Klas User akan menginstansiasi Klas Account dan akan memanfaatkan operasi-operasi yang dimilikinya.

User
- *Private Account : Account*
+ *Login (username : String, password : String) : String* + *Logout ( ) : String*

4.4.2.8 Penjelasan *Class Diagram* untuk Klas Administrator

Klas ini berguna untuk menginstansiasi klas Account dan akan memanfaatkan operasi-operasi yang dimilikinya, serta memfasilitasi pencatatan user yang diizinkan untuk menggunakan sistem kedalam basis data.

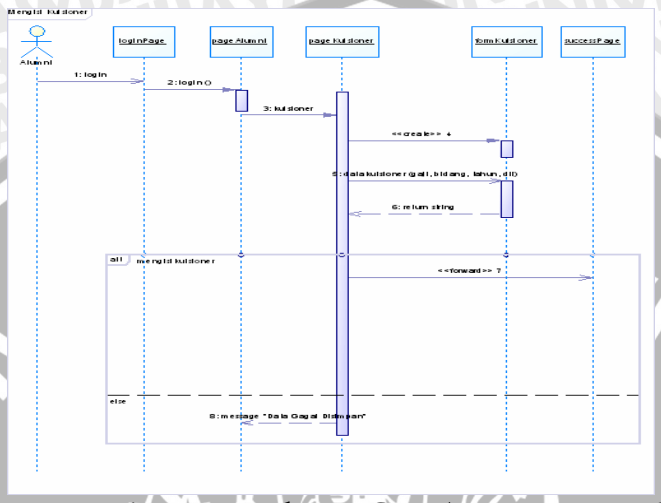
Administrator
- *Private Account : Account*
+ *melihat_account ( ) : String* + *menambah_account ( ) : String* + *mengganti_account ( ) : String* + *menghapus_account ( ) : String* + *menghapus_comment ( ) : String* + *melihat_fordis ( ) : String*

4.4.3 *Sequence Diagram*

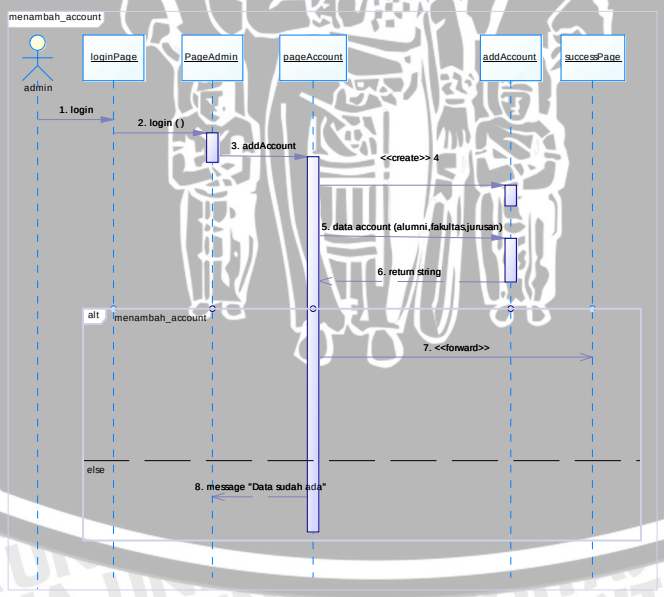
Hubungan antar klas yang digambarkan pada diagram klas termasuk dalam pemodelan statis, tanpa disertai dengan gambaran aliran proses atau data antarklas yang satu dengan klas yang lain. *Sequence diagram* berguna untuk menampikan aliran jalannya proses yang ditunjukkan dengan interaksi antar objek atau klas, dan disusun berdasarkan urutan waktu. *Sequence diagram* dirancang dengan

mengambil acuan pada *use case* serta operasi – operasi dalam klas yang menjadi implementasi dari fungsionalitas yang digambarkan pada *use case* tersebut.

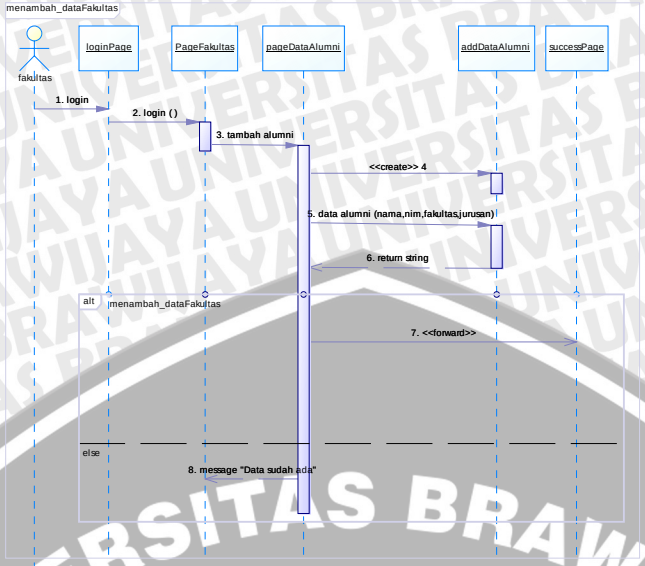
Sub bab 4.4.3 ini tidak memodelkan keseluruhan *sequence diagram* untuk tiap *use case*, akan tetapi diambil contoh *sequence diagram* untuk mengisi kuisioner, menambah *account*, dan menambah data fakultas.



Gambar 4.9 *Sequence Diagram* untuk *Use Case* Anggota *Tracer Study* pada Klas Alumni dengan Operasi Mengisi Kuisioner
[Analisis]



Gambar 4.10 *Sequence Diagram* untuk *Use Case* Administrasi Sistem *Tracer Study* pada Klas Admin dengan Operasi Menambah Account
[Analisis]



Gambar 4.11 *Sequence Diagram* untuk *Use Case* Kegiatan Pengelolaan Data Fakultas Sistem *Tracer Study* dengan Operasi Menambah Data Fakultas [Analisis]

4.5 Perancangan Basis Data

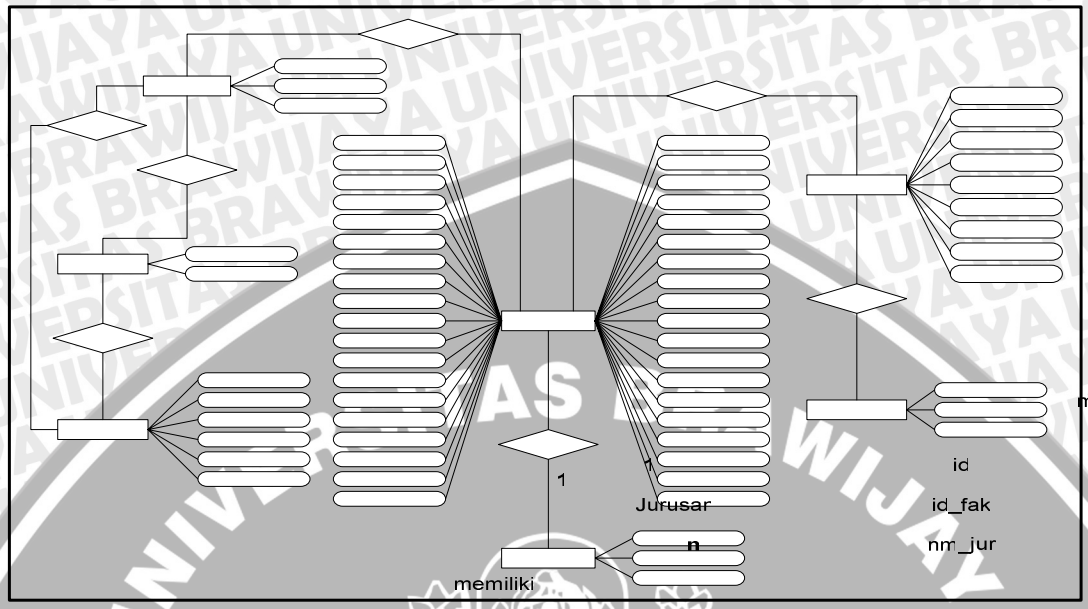
Sistem aplikasi yang akan dikembangkan ini memerlukan suatu basis data yang berfungsi untuk menampung data-data yang terkait. Perancangan basis data dilakukan agar basis data dapat efisien dalam penggunaan ruang penyimpanan, cepat dalam pengaksesan, dan mudah dalam pemanipulasian data. Perancangan basis data dapat dilakukan dengan menggambarkan diagram hubungan antar entitas atau biasa dikenal dengan *Entity –Relationship Diagram* (ER-Diagram), dan normalisasi. Normalisasi merupakan teknik dalam mengelompokkan atribut dari suatu relasi sehingga terbentuk struktur yang baik. Dalam E-R Diagram akan tampak relasi antar entitas yang saling terkait.

4.5.1 *Entity Relationship Diagram (E-R Diagram)*

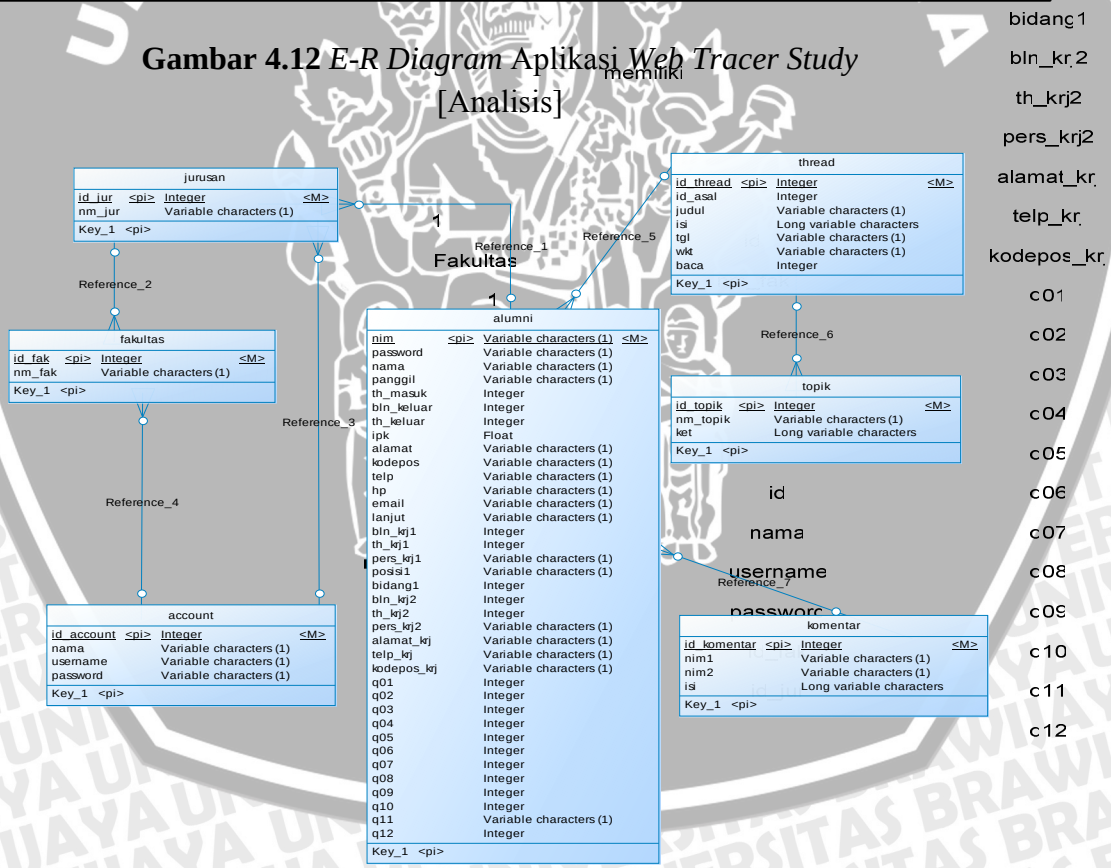
Entity Relationship Diagram (E-R Diagram) ini digunakan untuk menggambarkan hubungan entitas satu dengan lainnya dengan memperlihatkan hubungan antar *key* untuk berelasi antar tabel. Diagram Entitas Relasional dari aplikasi *Web Tracer Study* digambarkan pada gambar 4.9.

Diagram Relasi Antar Tabel disajikan untuk menampilkan relasi tabel-tabel yang digunakan dalam sistem ini. Diagram Relasi Antar Tabel digambarkan

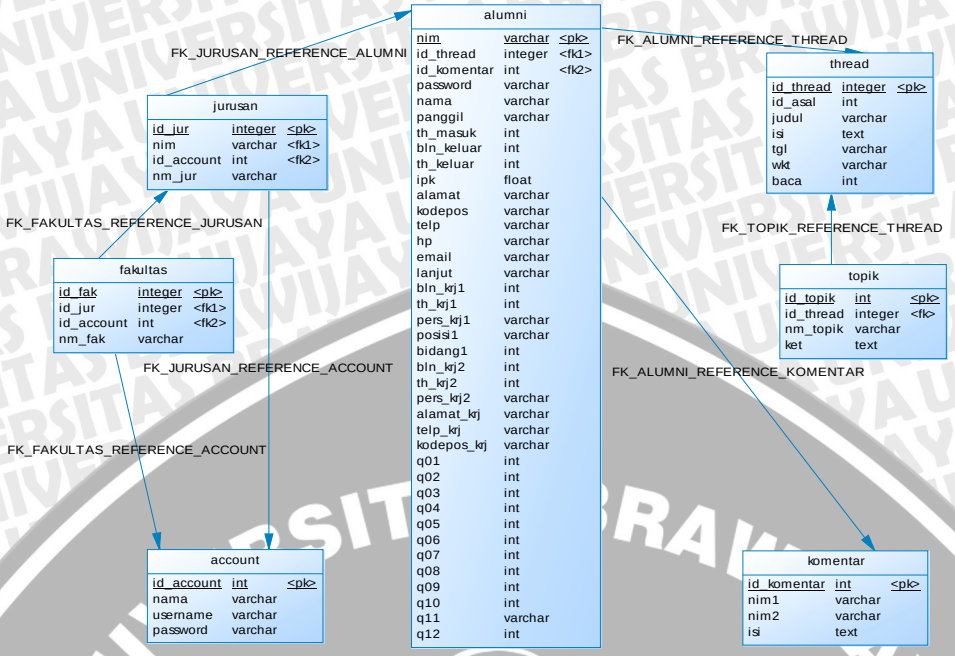
melalui *Conceptual Data Model* pada gambar 4.10 dan *Physical Data Model* pada gambar 4.12.



Gambar 4.12 *E-R Diagram* Aplikasi *Web Tracer Study* [Analisis]



Gambar 4.13 *Conceptual Data Model* dari *Web* Penelusuran Data Alumni UB [Analisis]



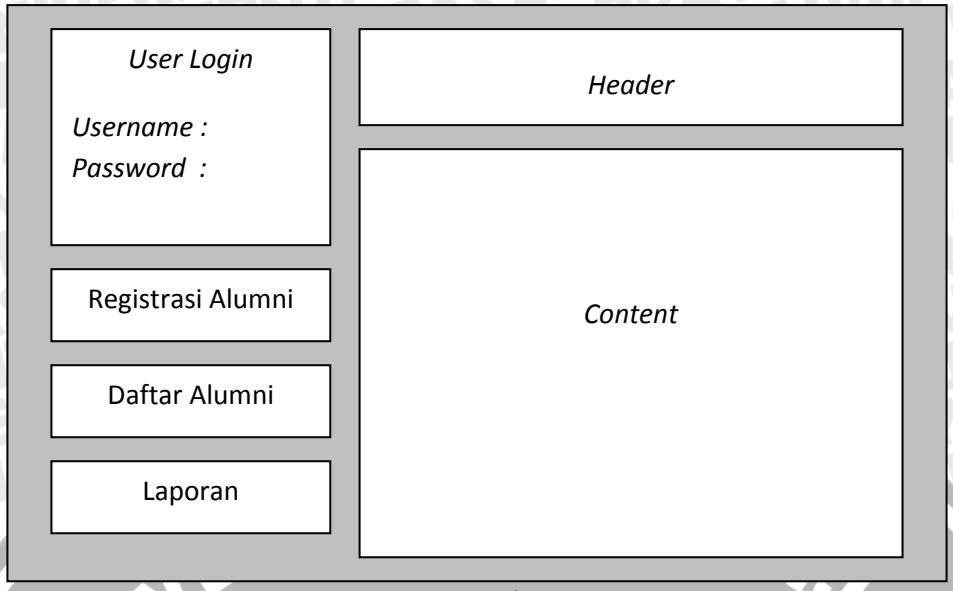
Gambar 4.14 *Physical Data Model* untuk *Web* Penelusuran Data Alumni UB [Analisis]

4.6 Perancangan Antarmuka Sistem

Perancangan antarmuka aplikasi *Web Tracer Study* terdiri atas delapan bagian, yaitu halaman *index,* halaman utama *administrator,* halaman utama fakultas, halaman utama jurusan, halaman registrasi alumni, halaman utama alumni, halaman data alumni, dan halaman laporan. Halaman utama merupakan halaman awal yang dapat diakses oleh siapapun. Setelah melakukan *login*, maka *user* dapat mengakses halaman yang sesuai dengan perannya masing-masing.

4.6.1 Perancangan Halaman *Index*

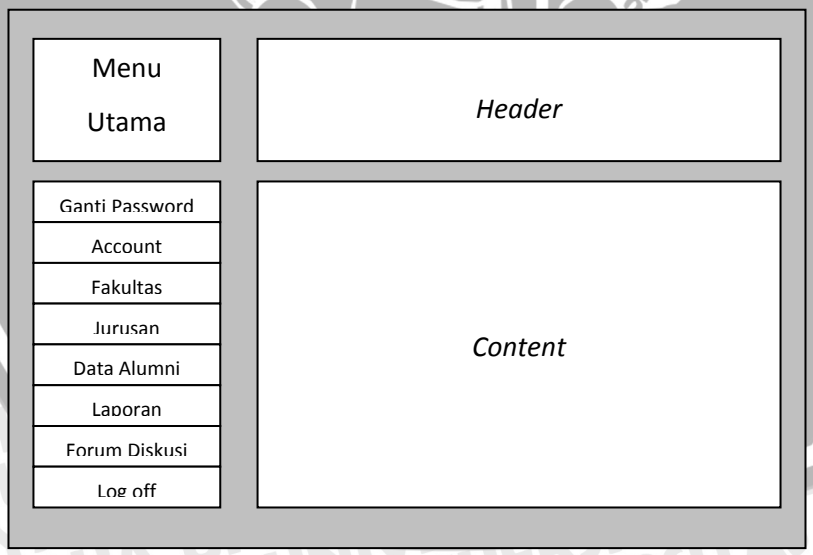
Halaman *index* merupakan halaman yang dapat diakses oleh siapapun. Pada halaman ini terdapat menu *login*, daftar alumni, laporan hasil rekapitulasi, serta registrasi alumni. Halaman ini juga menampilkan informasi umum *tracer study* UB.



Gambar 4.15 Halaman *Index*
[Analisis]

4.6.2 Perancangan Halaman Utama *Administrator*

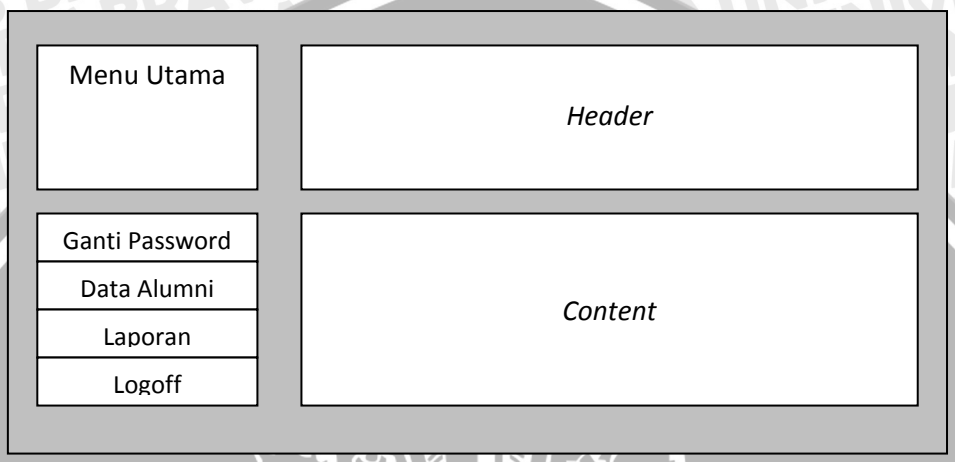
Halaman utama *Administrator* merupakan halaman yang hanya dapat diakses oleh *user* yang telah *login* sebagai *administrator* saja. Pada halaman ini terdapat menu ganti *password*, *account*, fakultas, jurusan, data alumni, laporan, forum diskusi, dan *log off*.



Gambar 4.16 Halaman Utama Administrator
[Analisis]

4.6.3 Perancangan Halaman Utama Fakultas

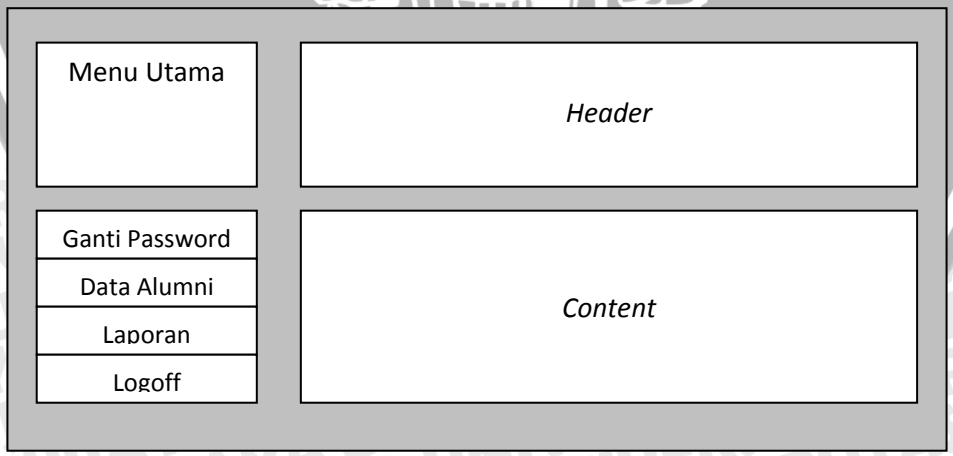
Halaman utama Fakultas adalah halaman yang hanya dapat diakses oleh *user* yang telah *login* sebagai Fakultas. Pada halaman ini terdapat menu untuk mengubah *password*, melihat profil alumni, melihat laporan hasil rekapitulasi kuisioner, dan *log off*.



Gambar 4.17 Halaman Utama Fakultas
[Analisis]

4.6.4 Perancangan Halaman Utama Jurusan

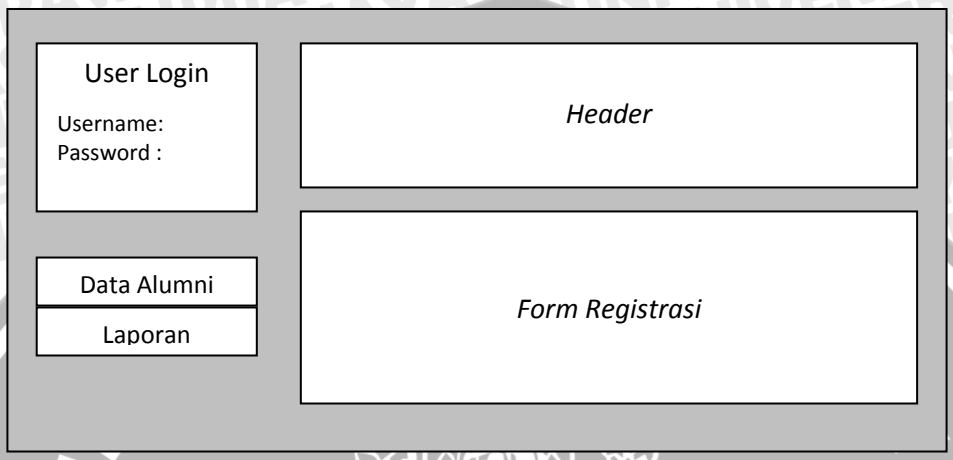
Halaman utama Jurusan hanya dapat diakses oleh *user* yang telah *login* sebagai Jurusan. Pada halaman ini terdapat menu utama yaitu Ganti *Password*, Data Alumni, Laporan, dan *Logoff*.



Gambar 4.18 Halaman Utama Jurusan
[Analisis]

4.6.5 Perancangan Halaman Registrasi Alumni

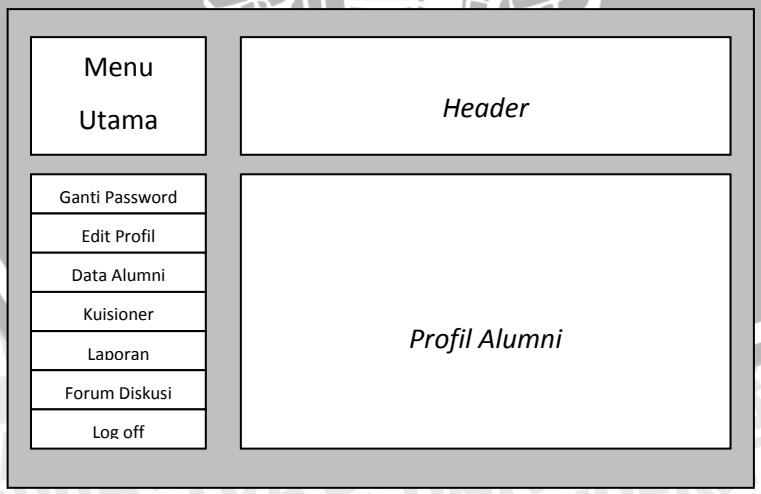
Halaman registrasi alumni merupakan halaman yang bisa diakses *user* eksternal yang ingin melakukan registrasi sebagai alumni terdaftar. Pada halaman ini menampilkan *form* registrasi alumni yang harus diisi oleh alumni yang ingin mendaftar.



Gambar 4.19 Halaman Registrasi Alumni
[Analisis]

4.6.6 Perancangan Halaman Utama Alumni

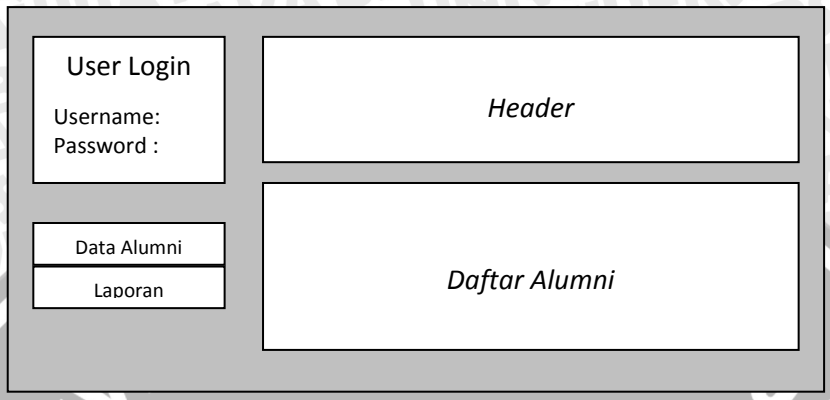
Halaman utama alumni merupakan halaman yang hanya dapat diakses oleh *user* yang telah melakukan *login* sebagai alumni. Pada halaman ini terdapat menu untuk mengganti *password*, mengedit profil, melihat profil alumni, mengisi kuisioner, melihat laporan, melihat forum diskusi, dan *log off*.



Gambar 4.20 Halaman Utama Alumni
[Analisis]

4.6.7 Perancangan Halaman Data Alumni

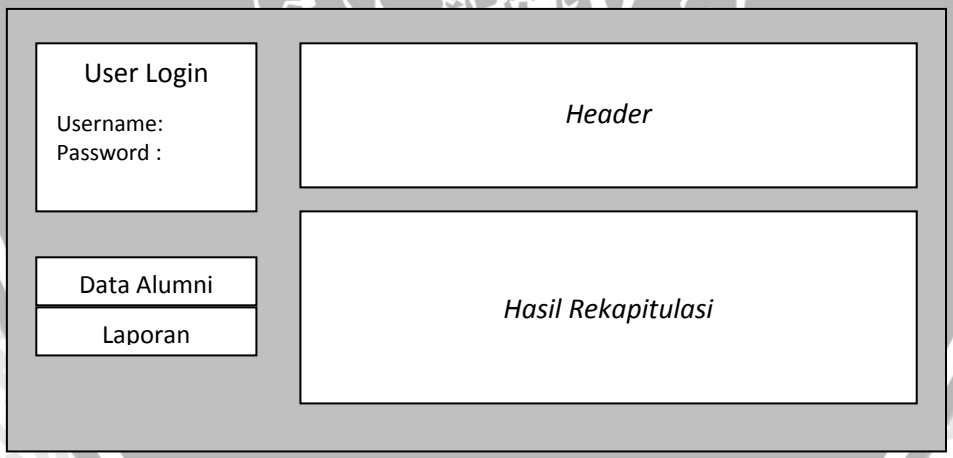
Halaman data alumni ini dapat diakses oleh semua *user*. Pada halaman ini terdapat nama-nama alumni yang telah terdaftar dan juga menu untuk melakukan pencarian profil alumni tertentu.



Gambar 4.21 Halaman Data Alumni
[Analisis]

4.6.8 Perancangan Halaman Laporan

Halaman laporan adalah halaman yang berisi hasil rekapitulasi kuisioner yang telah diisi oleh alumni terdaftar. Halaman ini dapat diakses oleh siapapun.

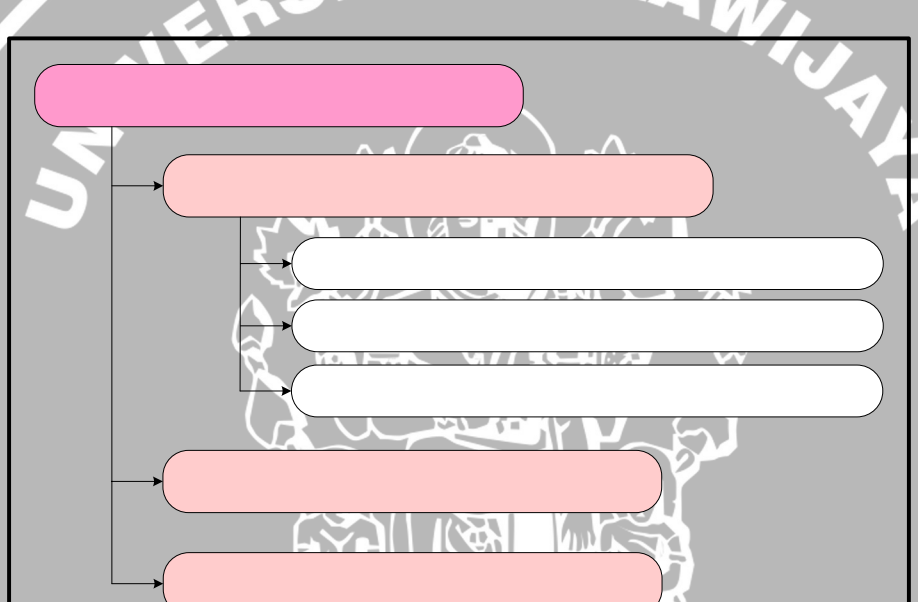


Gambar 4.22 Halaman Laporan
[Analisis]

BAB V

IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak sebelumnya. Pembahasan terdiri dari penjelasan implementasi perancangan basis data, dan implementasi perancangan perangkat lunak *Web* Penelusuran Data Alumni UB. Diagram alir implementasi perangkat lunak ditunjukkan dalam gambar 5.1 berikut.



Gambar 5.1 Diagram Alir Implementasi
Sumber: [Implementasi]

5.1 Implementasi Sistem

Hasil analisis kebutuhan dan perancangan yang telah diuraikan pada Bab 4 diimplementasikan menjadi sebuah sistem yang nyata agar bisa berfungsi sesuai dengan kebutuhan. Sistem yang dibuat akan diimplementasikan pada perangkat keras dan perangkat lunak dengan spesifikasi tertentu serta konfigurasi jaringan tertentu.

5. Implementasi

5.1 Implementasi Sistem

5.1.1 Spesifik

5.1.2 Spesifik

5.1.1 Spesifikasi Perangkat Keras (*Hardware*)

Untuk mengembangkan aplikasi *Web* Penelusuran Data Alumni UB, digunakan sebuah komputer yang berfungsi sebagai *server* dengan spesifikasi perangkat keras seperti terlihat dalam Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

4. Nama Komponen	5. Spesifikasi
6. Processor	7.Intel Core 2 Duo T5500 1.66 GHz
8. Memory	9.RAM 512 MB DDR2
10. Hardisk	11. ST9120822AS
12. Motherboard	13. TravelMate 6291
14. LAN Card	15. Broadcom NetLink (TM) Fast Ethernet
16. VGA Card	17. Mobile Intel(R) 945GM/GU Express Chipset Family (128 MB)

Sumber: [Implementasi]

5.1.2 Spesifikasi Perangkat Lunak (*Software*)

Untuk mengembangkan aplikasi *Web* Penelusuran Data Alumni UB, digunakan perangkat lunak dengan spesifikasi seperti terlihat dalam Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak

18. Perangkat Lunak	19. Spesifikasi
20. Sistem Operasi	21. Microsoft Windows XP version 5.1 SP2
22. Bahasa Pemrograman	23. PHP 5.2.5
24. Basis Data	25. MySQL version 5.0.51a
26. Web Server	27. Apache version 2.2.4 WIN32
28. Database Administrator Tool	29. phpMyAdmin version 2.11.4
30. Database Diagram Tool	31. Power Designer version 12.5
32. UML *Tool*	33. Power Designer version 12.5

Sumber: [Implementasi]

5.1.3 Spesifikasi Jaringan Komputer

Web Penelusuran Data Alumni UB ini diimplementasikan pada sebuah jaringan komputer lokal. Spesifikasi jaringan lokal *(Local Area Network)* tersebut dijelaskan melalui Tabel 5.3.

Tabel 5.3 Spesifikasi Jaringan Komputer

34.	35. Spesifikasi
36. Topologi Fisik	37. Bintang (Star)
38. Topologi Logik	39. 10/100 Base T
40. LAN Card	41. Broadcom NetLink (TM) Fast Ethernet
42. Kabel Jaringan	43. UTP cat 5e Belden USA
44. Switch	45. 3Com 3C16470 16 port 10/100

Sumber: [Implementasi]

Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berkut :

- *Web Browser* yang digunakan pada komputer *admin* adalah Internet Explorer 6.0
- Komputer server dan komputer *admin* dihubungkan oleh jaringan lokal.

5.2 Implementasi Perancangan Basis Data

Aplikasi ini dirancang untuk dapat terhubung ke *server* basis data MySQL. Implementasi perancangan basis data `skripsi` dilakukan sesuai dengan *Entity Relationship Diagram*. Implementasi perancangan basis data menggunakan *query* SQL. *Query* SQL digunakan untuk mengimplementasikan rancangan basis data ke dalam sistem basis data MySQL.

5.2.1 DDL untuk Membuat Basis Data `tracer`

DDL yang digunakan untuk membuat basis data `tracer` adalah sebagai

```
CREATE DATABASE db_tracer;
```

5.2.2 DDL untuk Membuat Tabel `account`

DDL yang digunakan untuk membuat table `account` adalah sebagai berikut:

```
CREATE TABLE account (
id_account int(3) UNSIGNED NOT NULL AUTO_INCREMENT,
nama varchar(20) NOT NULL,
username varchar(10) NOT NULL,
password varchar(10) NOT NULL,
id_fak int(3) UNSIGNED NOT NULL,
id_jur int(3) UNSIGNED NOT NULL,
PRIMARY KEY (id_account),
FOREIGN KEY (id_fak) REFERENCES fakultas (id_fak),
FOREIGN KEY (id_jur) REFERENCES jurusan (id_jur)
) ENGINE=MyISAM;
```

5.2.3 DDL untuk Membuat Tabel `alumni`

DDL yang digunakan untuk membentuk tabel `alumni` adalah sebagai berikut:

```
CREATE TABLE alumni (
nim varchar(5) NOT NULL,
password varchar(10) NOT NULL,
nama varchar(30) NOT NULL,
panggil varchar(10) NOT NULL,
id_jur int(3) UNSIGNED NOT NULL,
th_masuk int(4) UNSIGNED NOT NULL,
bl_keluar int(2) UNSIGNED NOT NULL,
th_keluar int(4) UNSIGNED NOT NULL,
ipk float NOT NULL,
alamat varchar(60) NOT NULL,
kodepos varchar(5) NOT NULL,
telp varchar(10) NOT NULL,
hp varchar(15) NOT NULL,
email varchar(30) NOT NULL,
lanjut varchar(20) NOT NULL,
bln_krj1 int(2) UNSIGNED NOT NULL,
th_krj1 int(4) UNSIGNED NOT NULL,
pers_krj1 varchar(20) NOT NULL,
posisi1 varchar(20) NOT NULL,
bidang1 int(1) UNSIGNED NOT NULL,
bln_krj2 int(2) UNSIGNED NOT NULL,
th_krj2 int(4) UNSIGNED NOT NULL,
pers_krj2 varchar(20) NOT NULL,
alamat_krj varchar(60) NOT NULL,
telp_krj varchar(10) NOT NULL,
kodepos_krj varchar(5) NOT NULL,
q01 int(1) UNSIGNED NULL,
q02 int(1) UNSIGNED NULL,
q03 int(1) UNSIGNED NULL,
q04 int(1) UNSIGNED NULL,
q05 int(1) UNSIGNED NULL,
q06 int(1) UNSIGNED NULL,
q07 int(1) UNSIGNED NULL,
q08 int(1) UNSIGNED NULL,
q09 int(1) UNSIGNED NULL,
q10 int(1) UNSIGNED NULL,
```

```
q11 varchar(20) NULL,
q12 int(1) UNSIGNED NULL,
PRIMARY KEY (nim),
FOREIGN KEY (id_jur) REFERENCES jurusan (id_jur)
) ENGINE=MyISAM;
```

5.2.4 DDL untuk Membuat Tabel `fakultas`

DDL yang digunakan untuk membentuk tabel `fakultas` adalah sebagai berikut:

```
id_fak int(3) UNSIGNED NOT NULL AUTO INCREMENT,
nm_fak varchar(20) NOT NULL,
PRIMARY KEY (id_fak)
) ENGINE=MyISAM;
```

5.2.5 DDL untuk Membuat Tabel `jurusan`

DDL yang digunakan untuk membentuk tabel `jurusan` adalah sebagai berikut:

```
id_jur int(3) UNSIGNED NOT NULL AUTO INCREMENT,
id_fak int(3) UNSIGNED NOT NULL,
nm_jur varchar(20) NOT NULL,
PRIMARY KEY (id_jur),
FOREIGN KEY (id_fak) REFERENCES fakultas (id_fak)
) ENGINE=MyISAM;
```

5.2.6 DDL untuk Membuat Tabel `thread`

DDL yang digunakan untuk membentuk tabel `thread` adalah sebagai berikut:

```
id_thread int(3) UNSIGNED NOT NULL AUTO INCREMENT,
id_topik int(3) UNSIGNED NOT NULL,
nim varchar(5) NOT NULL,
id_asal int(4) UNSIGNED NOT NULL,
judul varchar(20) NOT NULL,
isi text NOT NULL,
tgl varchar(10) NOT NULL,
wkt varchar(8) NOT NULL,
baca int(5) UNSIGNED NOT NULL,
PRIMARY KEY (id_thread),
FOREIGN KEY (id_topik) REFERENCES topik (id_topik),
FOREIGN KEY (nim) REFERENCES alumni (nim)
) ENGINE=MyISAM;
```

5.2.7 DDL untuk Membuat Tabel `topik`

DDL yang digunakan untuk membentuk tabel `topik` adalah sebagai berikut:

```
id_topik int(4) UNSIGNED NOT NULL AUTO INCREMENT,
nm_topik varchar(20) NOT NULL,
ket text NOT NULL,
PRIMARY KEY (id_topik),
) ENGINE=MyISAM;
```

5.2.8 DDL untuk Membuat Tabel `komentar`

DDL yang digunakan untuk membentuk tabel `komentar` adalah sebagai berikut:

```
id_komentar int(3) UNSIGNED NOT NULL AUTO INCREMENT,
nim int(3) UNSIGNED NOT NULL,
nim int(3) UNSIGNED NOT NULL,
isi varchar(20) NOT NULL,
PRIMARY KEY (id_jur),
FOREIGN KEY (id_fak) REFERENCES fakultas (id_fak)
) ENGINE=MyISAM;
```

5.3 Implementasi Antarmuka dan Algoritma

Subbab implementasi antarmuka dan algoritma ini akan menjelaskan semua implementasi antarmuka yang diperoleh dari daftar kebutuhan fungsional pada tabel 4.2. Subbab ini membahas 11 implementasi antarmuka yang ada pada tabel 4.2.

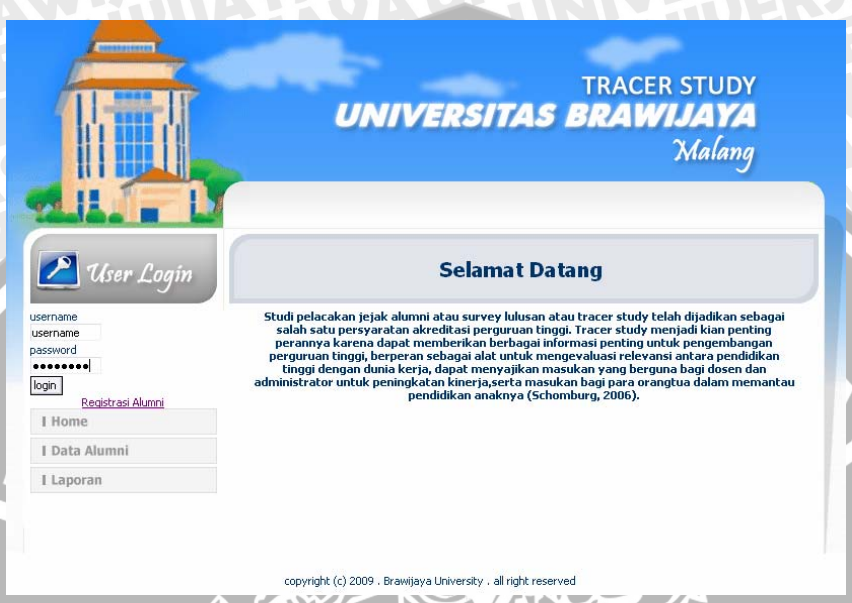
5.3.1 Implementasi Antarmuka Kebutuhan Fungsional *Login*

Implementasi dilakukan dengan menyediakan dua buah *text field* untuk masing-masing *username* dan *password*, dan satu tombol *login*.

Pengguna akan *login* sesuai peran yang dimiliki, jika pengguna berhasil melakukan *login* maka *user* akan memiliki hak tertentu sesuai peran yang

dimiliki. Dan selanjutnya pengguna akan menuju halaman utama terkait dengan pengguna yang melakukan *login*.

Implementasi kebutuhan Fungsional *login* dapat dilihat pada gambar 5.2 dibawah ini.



Gambar 5.2 Halaman untuk Melakukan Login
Sumber: [Implementasi]

Algoritma pada tombol "login" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

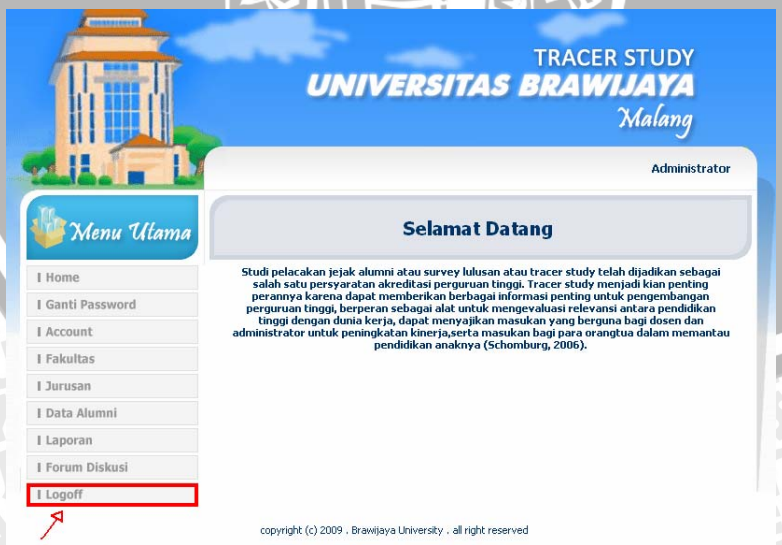
```
FUNCTION Login

VARIABLE user=VARIABLE _POST['user']
VARIABLE pass=VARIABLE _POST['pass']


IF FUNCTION not isset BASED ON VARIABLE user AND FUNCTION not isset BASED ON
VARIABLE pass THEN
   IF FUNCTION isset BASED ON VARIABLE _POST['ses'] THEN
      FUNCTION session_unset
      FUNCTION session_destroy
   ENDIF
ELSE
   VARIABLE user=FUNCTION trim BASED ON VARIABLE _POST['user']
   VARIABLE pass=FUNCTION trim BASED ON VARIABLE _POST['pass']
   VARIABLE str[1] = "select * from account where username='VARIABLE user' and
password='VARIABLE pass'"
   VARIABLE str[2] = "select * from alumni where nim='VARIABLE user' and
password='VARIABLE pass'"
   VARIABLE cek_login=1

   WHILE VARIABLE cek_login<3
      BEGIN
         VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str[VARIABLE
cek_login] OR FUNCTION die BASED ON FUNCTION mysql_error
        VARIABLE cek = FUNCTION mysql_num_rows BASED ON VARIABLE qry
        VARIABLE row = FUNCTION mysql_fetch_array BASED ON VARIABLE qry
IF VARIABLE cek=1 THEN
            IF VARIABLE cek_login=1 THEN
              VARIABLE id_jur=VARIABLE row['id_jur']
              VARIABLE id_fak=VARIABLE row['id_fak']
              IF VARIABLE id_jur=0 AND VARIABLE id_fak=0 VARIABLE hak=1 THEN

               ELSE IF VARIABLE id_fak>0 AND VARIABLE id_jur=0 THEN VARIABLE
hak=2
```

```
ELSE IF VARIABLE id_jur>0 AND VARIABLE id_fak=0 THEN VARIABLE hak=2
            ENDIF
            ELSE
                  VARIABLE hak=3
             END
            VARIABLE finish=true

            VARIABLE nama=VARIABLE row['nama']
            VARIABLE id=VARIABLE row[0]
            VARIABLE _SESSION['id']=VARIABLE id
            VARIABLE _SESSION['user']=VARIABLE user
            VARIABLE _SESSION['nama']=VARIABLE nama
            VARIABLE _SESSION['hak']=VARIABLE hak
             FUNCTION break
         ENDIF
         VARIABLE cek_login++
   END
END
```

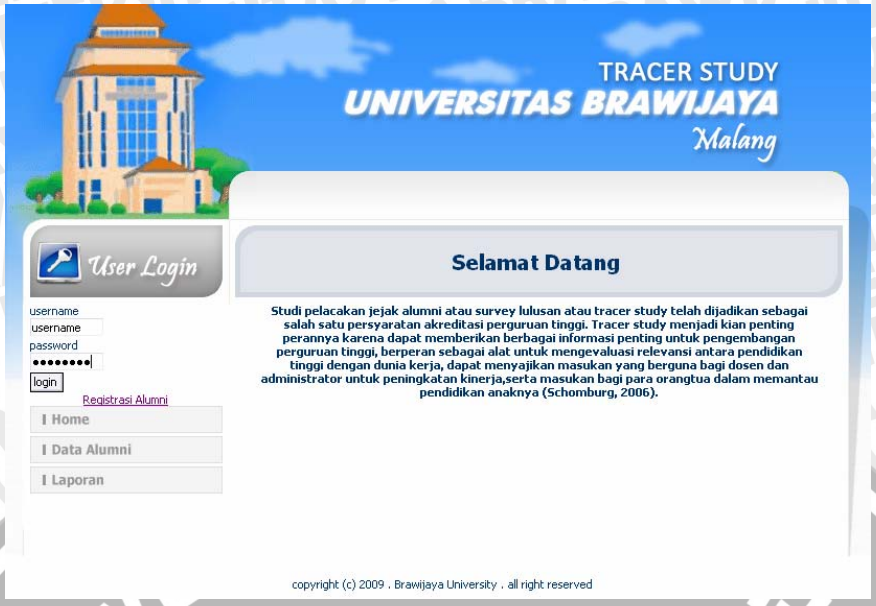
5.3.2 Implementasi Antarmuka Kebutuhan Fungsional Logout

Implementasi Kebutuhan Fungsional *logout* dilakukan dengan menyediakan satu buah tombol yang hanya muncul apabila telah melakukan login. Setelah melakukan *logout*, pengguna akan kembali ke halaman index.

Implementasi kebutuhan Fungsional *logout* dapat dilihat pada gambar 5.3 dan 5.4 dibawah ini.



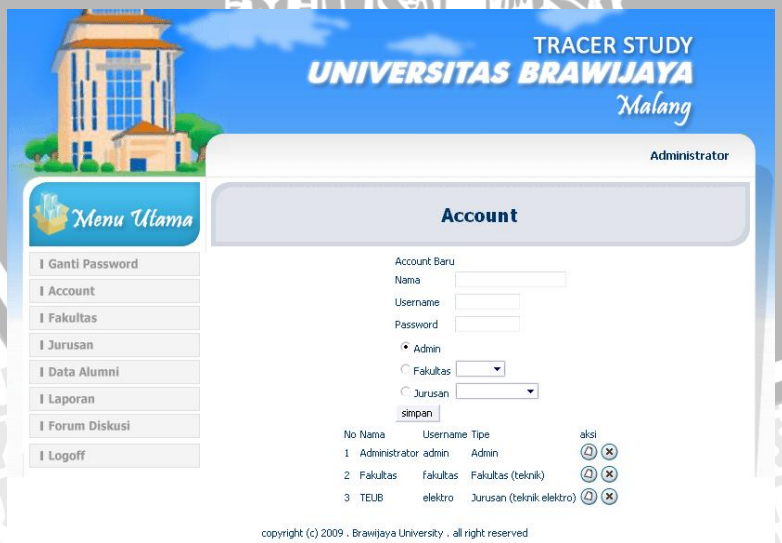
Gambar 5.3 Implementasi Antarmuka untuk Logout
Sumber: [Implementasi]



Gambar 5.4 Halaman setelah Melakukan Logout
Sumber: [Implementasi]

5.3.3 Implementasi Antarmuka Kebutuhan Fungsional Menambah, Melihat, Mengganti, dan Menghapus Account

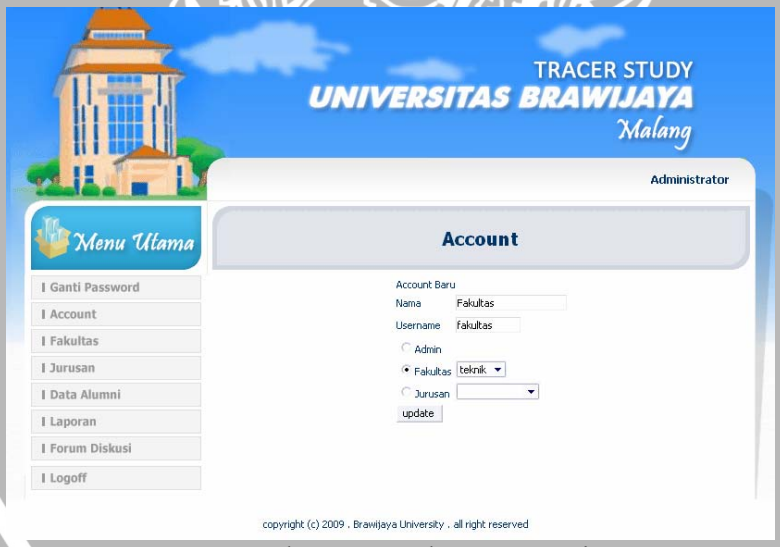
Untuk mengakses halaman ini pengguna masuk sebagai Admin. Kemudian pengguna akan menemukan beberapa sub menu, dimana salah satunya adalah sub menu Account. Pada submenu ini, user bisa memilih untuk melakukan aktivitas menambah, melihat, mengganti, ataupun menghapus *account*.



Gambar 5.5 Halaman untuk Melihat dan Menambah Account
Sumber: [Implementasi]

Algoritma pada tombol "simpan" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

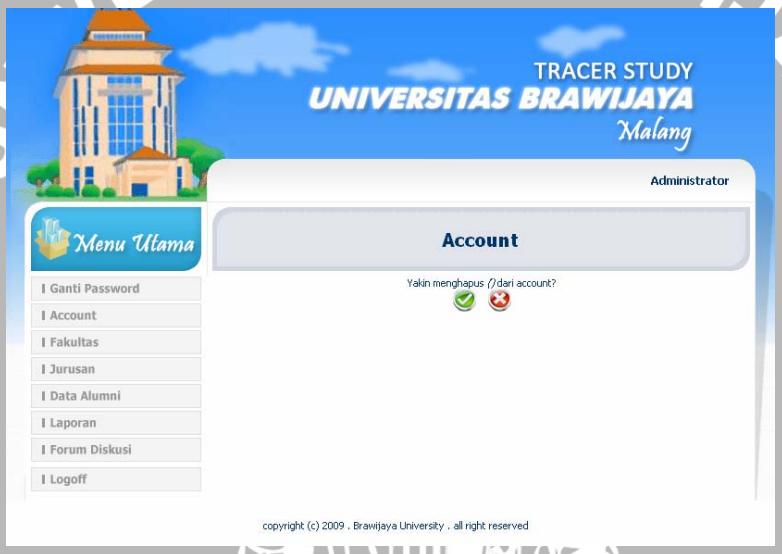
```
FUNCTION Simpan

    VARIABLE user=VARIABLE _POST['user']
    VARIABLE str = "select username from account where username='VARIABLE user'"
    VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON FUNCTION mysql_error
    VARIABLE cek = FUNCTION mysql_num_rows VARIABLE qry

    IF VARIABLE cek>0 echo "Account dengan username VARIABLE user sudah ada!!"
THEN

    ELSE
        VARIABLE nama=VARIABLE _POST['nama']
        VARIABLE pass=VARIABLE _POST['pass']
       VARIABLE id_fak=VARIABLE _POST['id_fak']
       VARIABLE id_jur=VARIABLE _POST['id_jur']

        IF VARIABLE _POST[tipe]='1' THEN VARIABLE id_fak=0 VARIABLE id_jur=0
       ELSE IF VARIABLE _POST[tipe]='2' THEN VARIABLE id_jur=0
       ELSE IF VARIABLE _POST[tipe]='3' THEN VARIABLE id_fak=0
       VARIABLE str = "insert into Account set nama='VARIABLE
nama',username='VARIABLE user',password='VARIABLE pass',id_jur=VARIABLE
id_jur,id_fak=VARIABLE id_fak"
       VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON mysql_error
       echo "Account sudah tersimpan..."
    ENDIF
```



Gambar 5.6 Halaman untuk Mengganti Account
Sumber: [Implementasi]

Algoritma pada tombol "update" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION Update

    VARIABLE id=VARIABLE _GET['id']
    VARIABLE nama=VARIABLE _POST['nama']
    VARIABLE user=VARIABLE _POST['user']
    VARIABLE id_fak=VARIABLE _POST['id_fak']
    VARIABLE id_jur=VARIABLE _POST['id_jur']

    IF VARIABLE _POST[tipe]='1' THEN VARIABLE id_fak=0 AND VARIABLE id_jur=0

    ELSE IF VARIABLE _POST[tipe]='2' THEN VARIABLE id_jur=0

    ELSE IF VARIABLE _POST[tipe]='3' THEN VARIABLE id_fak=0
            VARIABLE str = "update account set nama='VARIABLE
nama',username='VARIABLE user',id_jur=VARIABLE id_jur,id_fak=VARIABLE id_fak where
id='VARIABLE id'"
           VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION
die BASED ON FUNCTION mysql_error
           echo "Account sudah terupdate..."

END
```



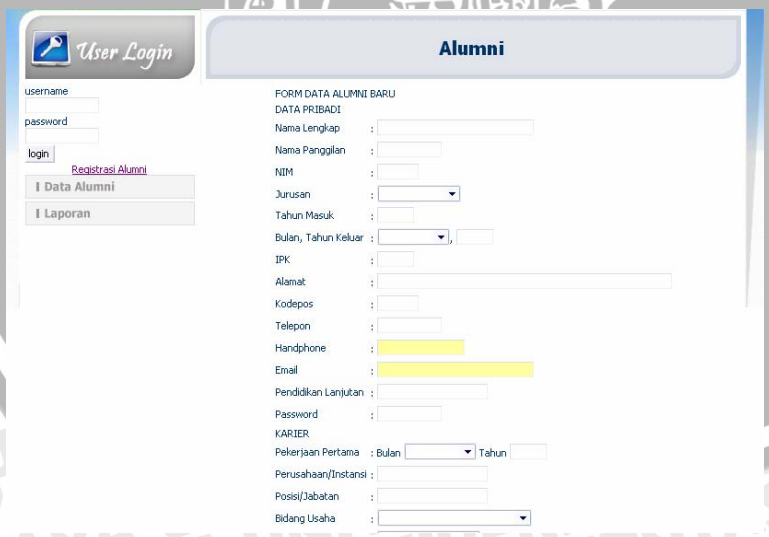
Gambar 5.7 Halaman untuk Menghapus Account
Sumber: [Implementasi]

Algoritma pada tombol “hapus” diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION confirmHapus

        VARIABLE id=VARIABLE _GET['id']
        VARIABLE str = "select nama,username from account where id=VARIABLE id"
        VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON FUNCTION mysql_error
        VARIABLE row = FUNCTION mysql_fetch_array BASED ON VARIABLE qry
        VARIABLE nama = VARIABLE row[nm_jur]
        echo "Yakin menghapus  <i>VARIABLE username (VARIABLE nama)</i> dari
account?<br>"
        echo "
                <a href='account.php?act=6&id=VARIABLE id'><img
src='images/yes.gif' border=0 alt='ya'></a>   
                <a href='account.php'><img src='images/no.gif' border=0
alt='tidak'></a>"
END

FUNCTION Hapus

        VARIABLE id=VARIABLE _GET['id']
        VARIABLE str = "delete from acount where id='VARIABLE id'"
        VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON FUNCTION mysql_error
        echo "account sudah terhapus...";
END
```

5.3.4 Implementasi Antarmuka untuk Kebutuhan Fungsional Mengisi Data Registrasi

Untuk menjadi anggota *tracer study*, *user* harus melakukan registrasi terlebih dahulu dengan mengisi *form* registrasi yang tersedia. *User* yang telah mengisi registrasi akan terdaftar sebagai alumni. *Form* pengisian data registrasi ditunjukkan pada Gambar 5.8 berikut ini.



Gambar 5.8 Halaman yang Menampilkan Form Pengisian Data Registrasi Alumni

Sumber: [Implementasi]

Algoritma pada tombol "simpan" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION Simpan

              VARIABLE str = "insert into alumni set nim='VARIABLE
_POST[nim]',nama='VARIABLE _POST[nama1]',panggil='VARIABLE _POST[nama2]',
                              tempat='VARIABLE _POST[tempat]',tgl='VARIABLE
_POST[tgl]',bln='VARIABLE _POST[bln]', thn='VARIABLE _POST[thn]',
                              password='VARIABLE _POST[pass]',id_jur='VARIABLE
_POST[id_jur]',th_masuk='VARIABLE _POST[th1]',
                              bln_keluar='VARIABLE
_POST[bln_keluar]',th_keluar='VARIABLE _POST[th_keluar]', ipk=VARIABLE _POST[ipk],
                              alamat='VARIABLE _POST[alamat]',kodepos='VARIABLE
_POST[pos]',telp='VARIABLE _POST[telp]',hp='VARIABLE _POST[hp]',
                              email='VARIABLE _POST[email]',bln_krj1='VARIABLE
_POST[bln_krj1]', th_krj1=VARIABLE _POST[th_krj1],
                              pers_krj1='VARIABLE
_POST[pers_krj1]',posisi1='VARIABLE _POST[posisi1]',bidang1='VARIABLE
_POST[bidang1]',
                              q06=VARIABLE _POST[q6],bln_krj2='VARIABLE
_POST[bln_krj2]', th_krj2=VARIABLE _POST[th_krj2],
                              pers_krj2='VARIABLE _POST[pers_krj2]',q11='VARIABLE
_POST[q11]',q04='VARIABLE _POST[q04]',
                              alamat_krj='VARIABLE
_POST[alamat_krj]',telp_krj='VARIABLE _POST[telp_krj]', kodepos_krj='VARIABLE
_POST[kodepos_krj]'"
              VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR
FUNCTION die BASED ON FUNCTION mysql_error
              echo "Data alumni sudah tersimpan..."
END
```

5.3.5 Implementasi Antarmuka untuk Kebutuhan Fungsional Melihat Data Profil

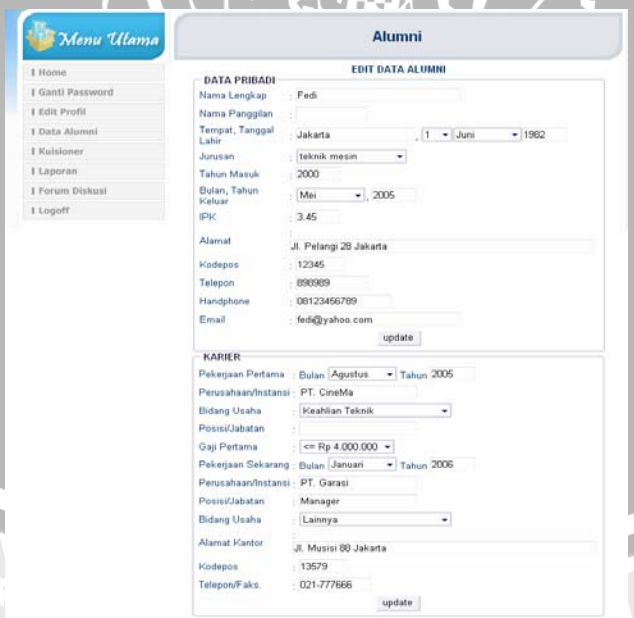
Implementasi antarmuka untuk kebutuhan fungsional melihat data profil ditunjukkan pada Gambar 5.9. Aktor tidak harus melakukan *login* untuk melihat data profil alumni. Untuk melihat data profil, pengguna dapat memilih menu Data Alumni.

Gambar 5.9 Implementasi Antarmuka untuk memilih menu Data Alumni
Sumber: [Implementasi]

Gambar 5.10 Hasil Implementasi Antarmuka untuk Melihat Profil Alumni
Sumber: [Implementasi]

5.3.6 Implementasi Antarmuka untuk Kebutuhan Fungsional Mengganti Data Profil

Implementasi antarmuka untuk mengganti data profil ditunjukkan pada Gambar 5.11. Pengguna yang telah *login* sebagai alumni dapat melakukan penggantian data profilnya dengan memilih menu Edit Profil.



Gambar 5.11 Implementasi Antarmuka untuk Mengedit Data Profil
Sumber: [Implementasi]

Algoritma pada tombol “update” diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION Update BASED ON VARIABLE id

       VARIABLE str = "update alumni set nama='VARIABLE
_POST[nama1]',panggil='VARIABLE _POST[nama2]',tempat='VARIABLE
_POST[tempat]',tgl='VARIABLE _POST[tgl]',bln='VARIABLE _POST[bln]', thn='VARIABLE
_POST[thn]',id_jur='VARIABLE _POST[id_jur]',th_masuk='VARIABLE
_POST[th1]',bln_keluar='VARIABLE _POST[bln_keluar]',th_keluar='VARIABLE
_POST[th_keluar]', ipk=VARIABLE _POST[ipk],alamat='VARIABLE
_POST[alamat]',kodepos='VARIABLE _POST[pos]',telp='VARIABLE
_POST[telp]',hp='VARIABLE _POST[hp]',email='VARIABLE
_POST[email]',bln_krj1='VARIABLE _POST[bln_krj1]', th_krj1=VARIABLE
_POST[th_krj1],pers_krj1='VARIABLE _POST[pers_krj1]',posisi1='VARIABLE
_POST[posisi1]',bidang1='VARIABLE _POST[bidang1]',q06=VARIABLE
_POST[q6],bln_krj2='VARIABLE _POST[bln_krj2]', th_krj2=VARIABLE
_POST[th_krj2],pers_krj2='VARIABLE _POST[pers_krj2]',q11='VARIABLE
_POST[q11]',q04='VARIABLE _POST[q04]',alamat_krj='VARIABLE
_POST[alamat_krj]',telp_krj='VARIABLE _POST[telp_krj]', kodepos_krj='VARIABLE
_POST[kodepos_krj]' where nim='VARIABLE id'";

       VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON FUNCTION mysql_error
              echo "Data Alumni sudah terupdate..."
END
```

5.3.7 Implementasi Antarmuka untuk Kebutuhan Fungsional Mengisi Kuisioner

Untuk dapat mengisi kuisioner, pengguna harus melakukan *login* sebagai alumni terlebih dahulu, kemudian memilih submenu Kuisioner pada submenu yang disediakan di sebelah kiri. Setelah memilih submenu Kuisioner, pengguna dapat mengisi form kuisioner dan menyimpannya. Implementasi antarmuka untuk mengisi kuisioner ditunjukkan pada Gambar 5.12 berikut.

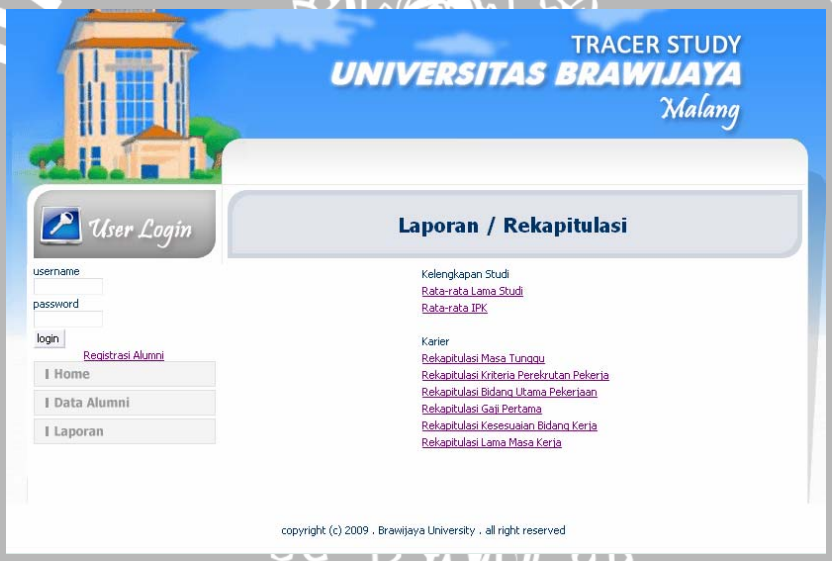
Gambar 5.12 Halaman yang Menampilkan Form untuk Mengisi Kuisioner
Sumber: [Implementasi]

Algoritma pada tombol “simpan” diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION simpanJawaban BASED ON VARIABLE nim

        VARIABLE str = "update alumni set q01='VARIABLE _POST[q01]',
                       q02='VARIABLE _POST[q02]',q03='VARIABLE _POST[q03]',
                       q04='VARIABLE _POST[q04]',q05='VARIABLE _POST[q05]',
                       q06='VARIABLE _POST[q06]',q07='VARIABLE _POST[q07]',
                       q08='VARIABLE _POST[q08]',q09='VARIABLE _POST[q09]',
                       q10='VARIABLE _POST[q10]',q11='VARIABLE _POST[q11]',
                       q12='VARIABLE _POST[q12]' WHERE nim='VARIABLE nim'"
        VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
         BASED ON FUNCTION mysql_error
        echo "Data Kuisioner Sudah Tersimpan..."
END
```

5.3.8 Implementasi Antarmuka untuk Kebutuhan Fungsional Melihat Laporan Rekapitulasi Hasil Kuisioner

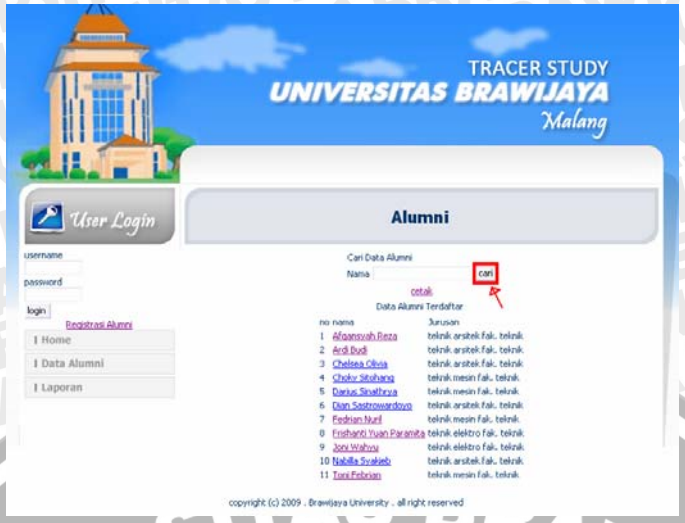
Untuk dapat mengakses halaman ini pengguna tidak perlu melakukan *login*. Pengguna dapat memilih submenu Laporan untuk melihat laporan rekapitulasi hasil kuisioner. Implementasi antarmuka untuk Melihat Laporan Rekapitulasi Hasil Kuisioner dapat dilihat pada Gambar 5.13 berikut.



Gambar 5.13 Implementasi Antarmuka untuk Melihat Laporan Hasil Kuisioner
Sumber: [Implementasi]

5.3.9 Implementasi Antarmuka untuk Kebutuhan Fungsional Mencari Profil Alumni

Untuk dapat mengakses halaman ini pengguna tidak perlu melakukan *login*. Pengguna dapat memilih submenu Data Alumni dan kemudian melakukan pencarian berdasarkan nama dan NIM alumni. Implementasi antarmuka untuk mencari profil alumni ditunjukkan pada Gambar 5.14 berikut ini.



Gambar 5.14 Implementasi Antarmuka untuk Mencari Profil Alumni
Sumber: [Implementasi]

Algoritma pada tombol "cari" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini:

```
FUNCTION Lihat BASED ON VARIABLE nm_cari,VARIABLE hak,VARIABLE id,VARIABLE cetak

        VARIABLE str = "select a.*,j.nm_jur,f.nm_fak from alumni a,jurusan j,
                        fakultas f WHERE a.id_jur=j.id and j.id_fak=f.id"
        IF VARIABLE hak=2 THEN
           VARIABLE s = "select id_fak,id_jur from account where id='VARIABLE
id'"
           VARIABLE q = FUNCTION mysql_query BASED ON VARIABLE s OR FUNCTION die
                         BASED ON FUNCTION mysql_error
           VARIABLE r = FUNCTION mysql_fetch_array BASED ON VARIABLE q

           IF VARIABLE r[0]>0 AND VARIABLE r[1]=0 THEN VARIABLE str = VARIABLE
str."
               AND f.id='VARIABLE r[0]'" THEN
           ELSE IF VARIABLE r[0]=0 AND VARIABLE r[1]>0 THEN
              VARIABLE str = VARIABLE str." AND j.id='VARIABLE r[1]'"
        ENDIF
         IF isset BASED ON VARIABLE nm_cari THEN VARIABLE str = VARIABLE str." AND
nama like '%VARIABLE nm_cari%'"
           VARIABLE str = VARIABLE str." order by a.nama ASC"
           VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION
die BASED ON FUNCTION mysql_error
           VARIABLE cek = FUNCTION mysql_num_rows BASED ON VARIABLE qry
           IF VARIABLE cek=0 THEN echo "Data Alumni Tidak Ada..." THEN
           ELSE
           VARIABLE hak=VARIABLE _SESSION['hak']
            ENDIF
```

```
IF NOT VARIABLE cetak THEN
        echo "
        <table><form method=post action='alumni.php'>
        <tr><td colspan=2>Cari Data Alumni</td></tr>
        <tr><td>Nama</td><td><input type='text' name='nm_cari' size=20
value='VARIABLE nm_cari'>  <input type='submit' value='cari'></td></tr>
        <tr><td colspan=2 align=center><a href='cetak.php?act=1&cari=VARIABLE
nm_cari'>cetak</a></td></tr>
        </form></table>";
        echo "<table><tr><td align=center>Data Alumni Terdaftar<td></tr></table>
              table><tr><td>no</td><td>nama</td><td>Jurusan</td>"
     IF VARIABLE hak=1 THEN echo "<td></td>";} echo "</tr>"
        VARIABLE i=0
     ENDIF
     WHILE VARIABLE row=FUNCTION mysql_fetch_array BASED ON VARIABLE qry
     BEGIN
     VARIABLE i++
     IF NOT VARIABLE cetak THEN
     echo "<tr><td>VARIABLE i</td><td><a href='alumni.php?act=7&id=VARIABLE
row[nim]'>VARIABLE row[nama]</a></td><td>VARIABLE row[nm_jur] fak. VARIABLE
row[nm_fak]</td>"

     ELSE echo "<tr><td>VARIABLE i</td><td>VARIABLE row[nama]</td><td>VARIABLE
              row[nm_jur] fak. VARIABLE row[nm_fak]</td>";
     IF VARIABLE hak=1 AND NOT VARIABLE cetak THEN
          echo "<td><a href='alumni.php?act=3&id=VARIABLE row[nim]'><img
src='images/edit.gif' alt='edit' border=0> <a href='alumni.php?act=5&id=VARIABLE
row[nim]'><img src='images/delete.gif' alt='hapus' border=0></td>"
     ENDIF
                echo "</tr>"
    ENDIF
                echo "</table>"
  END
END
```

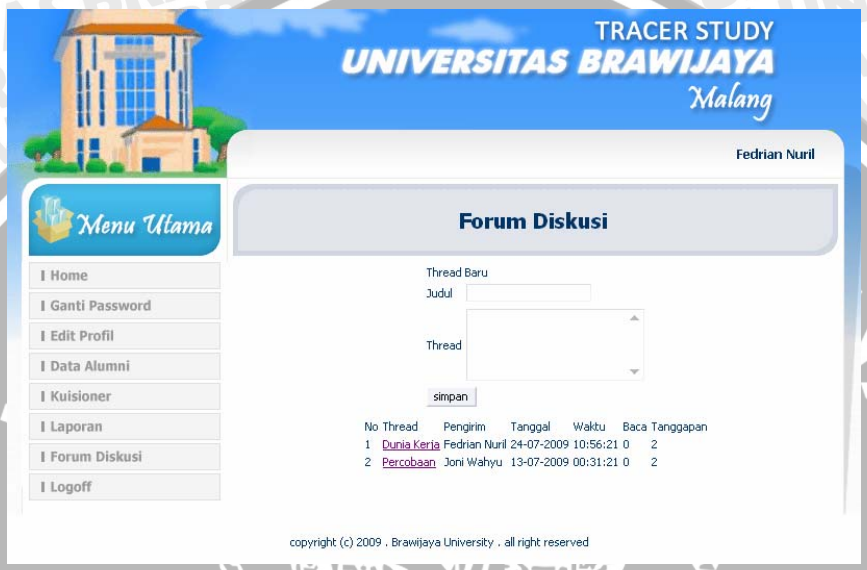
5.3.10 Implementasi Antarmuka untuk Kebutuhan Fungsional Melihat Forum Diskusi

Implementasi antarmuka untuk melihat forum diskusi ditunjukkan pada Gambar 5.15 berikut. Pengguna harus *login* sebagai alumni atau administrator untuk dapat mengakses halaman ini.

Gambar 5.15 Implementasi Antarmuka untuk Melihat Forum Diskusi
Sumber: [Implementasi]

5.3.11 Implementasi Antarmuka untuk Kebutuhan Fungsional Mengisi Forum Diskusi

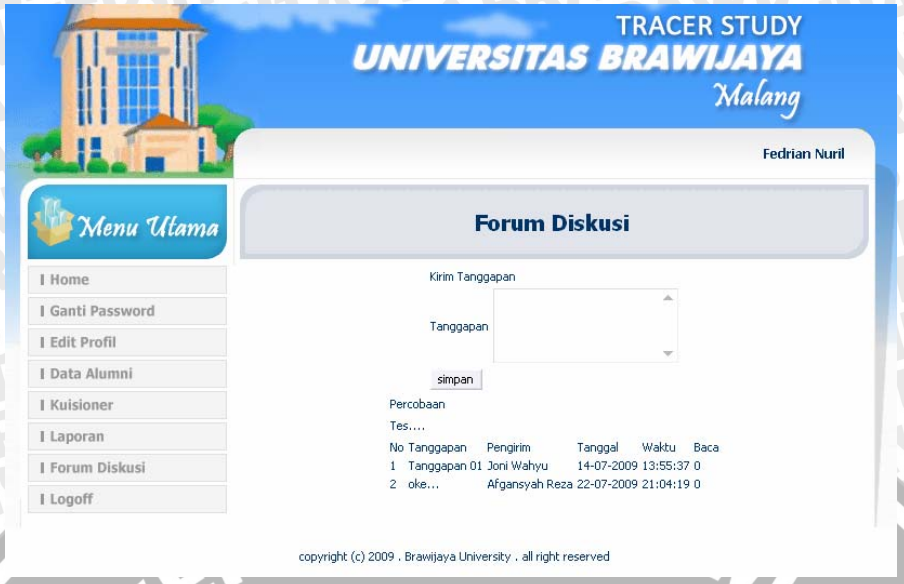
Untuk dapat mengakses halaman ini pengguna harus *login* sebagai Alumni terlebih dahulu. Alumni dapat membuat *thread* baru ataupun memberikan tanggapan pada *thread* yang sudah ada. Implementasi antarmuka untuk mengisi forum diskusi ditunjukkan pada Gambar 5.16 berikut ini.



Gambar 5.16 Implementasi Antarmuka untuk Membuat Thread Baru
Sumber: [Implementasi]

Algoritma pada tombol "simpan" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION simpanThread BASED ON VARIABLE id,VARIABLE jf,VARIABLE asal,VARIABLE
nim,VARIABLE isi,VARIABLE tgl,VARIABLE wkt

              VARIABLE str = "insert into thread set id_topik='VARIABLE
id',judul='VARIABLE jf',id_asal='VARIABLE asal',
                              nim='VARIABLE nim',isi='VARIABLE isi',tgl='VARIABLE
tgl',wkt='VARIABLE wkt'"
              VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR
FUNCTION die BASED ON FUNCTION mysql_error
              IF isset BASED ON VARIABLE asal THEN echo "Tanggapan sudah
tersimpan..."
              ELSE echo "Thread sudah tersimpan..."
               ENDIF
END
```



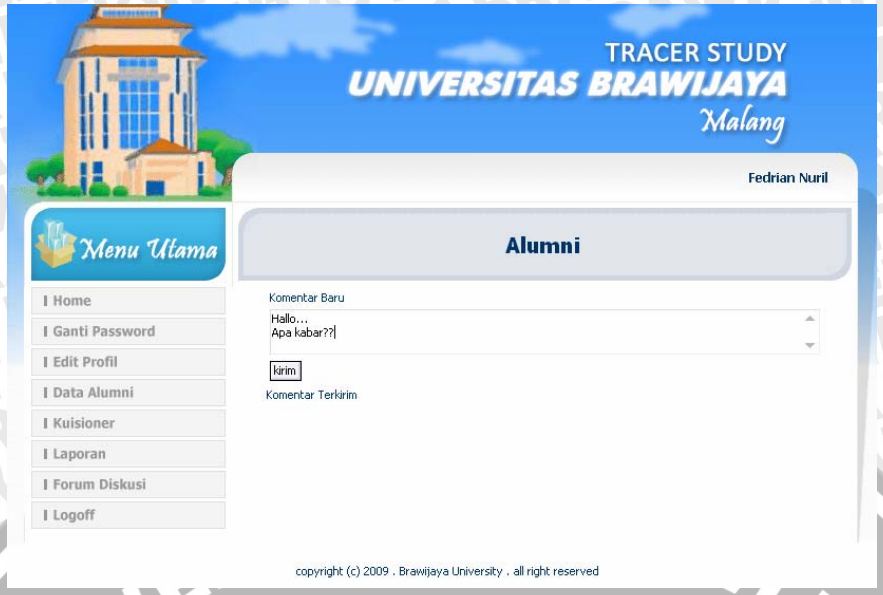
Gambar 5.17 Implementasi Antarmuka untuk Mengirim Tanggapan
Sumber: [Implementasi]

Algoritma pada tombol "simpan" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION simpanThread BASED ON VARIABLE id,VARIABLE jf,VARIABLE asal,VARIABLE
nim,VARIABLE isi,VARIABLE tgl,VARIABLE wkt

        VARIABLE str = "insert into thread set id_topik='VARIABLE
id',judul='VARIABLE jf',id_asal='VARIABLE asal',nim='VARIABLE nim',isi='VARIABLE
isi',tgl='VARIABLE tgl',wkt='VARIABLE wkt'"
        VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
                        BASED ON FUNCTION mysql_error
        IF isset BASED ON VARIABLE asal THEN echo "Tanggapan sudah tersimpan..."
        ENDIF
END
```

5.3.12 Implementasi Antarmuka untuk Kebutuhan Fungsional Menambah *Commment* Profil

Implementasi antarmuka untuk menambah *comment* profil hanya dapat diakses oleh pengguna yang telah *login* sebagai alumni. Alumni dapat memberikan *comment* pada profil alumni lain. Implementasi ini ditunjukkan pada Gambar 5.18 berikut ini.



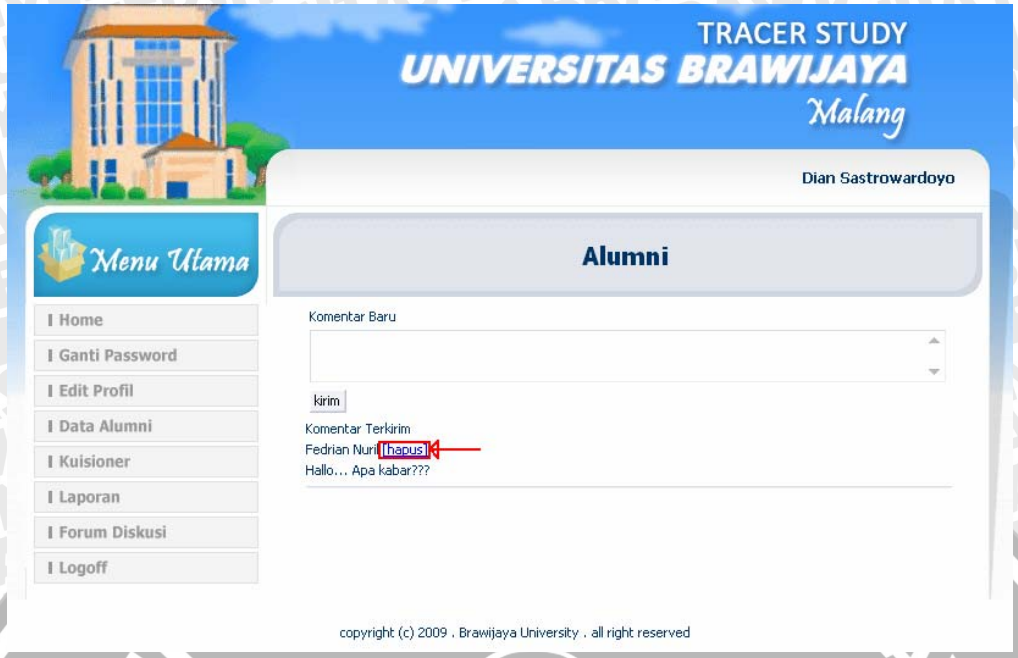
Gambar 5.18 Implementasi Antarmuka untuk Menambah Comment Profil
Sumber: [Implementasi]

Algoritma pada tombol "kirim" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION simpanKomentar VARIABLE id1,VARIABLE id2

       VARIABLE str = "insert into komentar set nim1='VARIABLE id1',nim2='VARIABLE
id2',isi='VARIABLE _POST[isi]'"
       VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die BASED
ON FUNCTION mysql_error
       echo "Komentar sudah terkirim..."

END
```

5.3.13 Implementasi Antarmuka untuk Kebutuhan Fungsional Menghapus *Comment* Profil

Untuk dapat mengakses halaman ini pengguna harus login sebagai alumni. Pengguna dapat menghapus comment dengan memilih tombol hapus. Implementasi antarmuka untuk menghapus comment ditunjukkan pada Gambar 5.19 berikut.



Gambar 5.19 Implementasi Antarmuka untuk Menghapus Comment Profil
Sumber: [Implementasi]

Algoritma pada tombol "hapus" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
FUNCTION hapusKomentar BASED ON VARIABLE id

       VARIABLE str = "delete from komentar where id='VARIABLE id'"
       VARIABLE qry = FUNCTION mysql_query BASED ON VARIABLE str OR FUNCTION die
BASED ON FUNCTION mysql_error
       echo "Komentar sudah terhapus..."

END
```

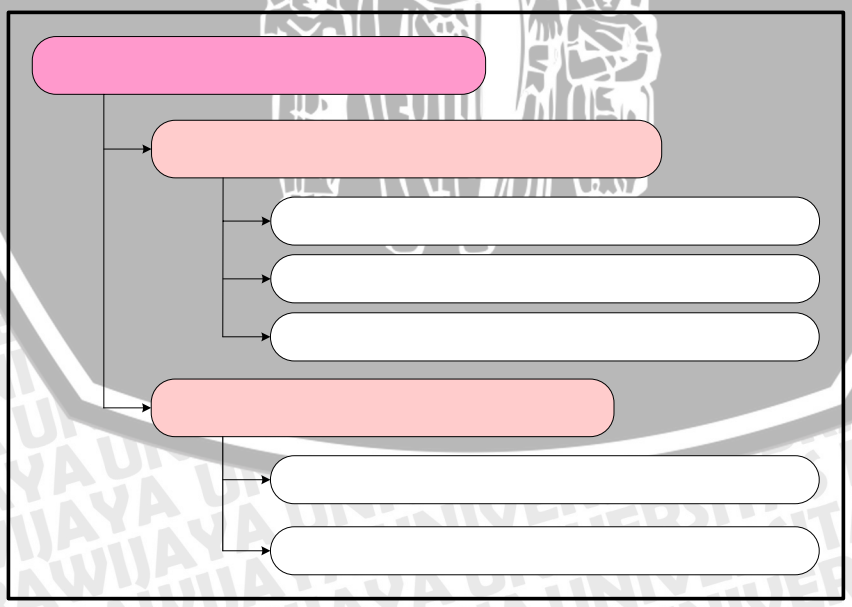
BAB VI

PENGUJIAN

Pada Bab ini dilakukan proses pengujian pada *Web* Penelusuran Data Alumni UB. Proses pengujian yang akan dilakukan terbagi menjadi dua bagian, yaitu pengujian perangkat lunak dan pengujian basis data.

Pengujian basis data meliputi pengujian rancangan basis data, pengujian koneksi basis data, dan pengujian waktu akses *query*. Pengujian rancangan basis data bertujuan untuk menguji apakah implementasi perancangan basis data telah sesuai dengan *Entity Relational Diagram*. Pengujian koneksi basis data dilakukan untuk memastikan bahwa komputer *client* dapat melakukan koneksi dengan *database* MySQL yang berada pada komputer *server*.

Pengujian Perangkat lunak terbagi menjadi dua tahap, yaitu pengujian unit dan pengujian integrasi. Pada pengujian unit akan digunakan teknik pengujian *white box (white box Testing)*. Pada pengujian integrasi akan digunakan teknik pengujian *black box (black box Testing)*. Gambar 6.1 menunjukkan langkah-langkah proses pengujian.



Gambar 6.1 Diagram Alir Pengujian
Sumber: [Pengujian]

6.1 Pengujian Basis Data

Pengujian basis data *web tracer study* ini meliputi tiga macam pengujian, yaitu pengujian rancangan basis data, pengujian koneksi basis data, dan pengujian waktu akses *query*.

6.1.1 Pengujian Rancangan Basis Data

Pengujian Rancangan basis data bertujuan untuk mengetahui kebenaran rancangan basis data yang telah dibuat. Pengujian perancangan basis data dilakukan dengan menggunakan perangkat lunak Sybase PowerDesigner 12.5.

Nama Kasus Uji : Rancangan Basis Data

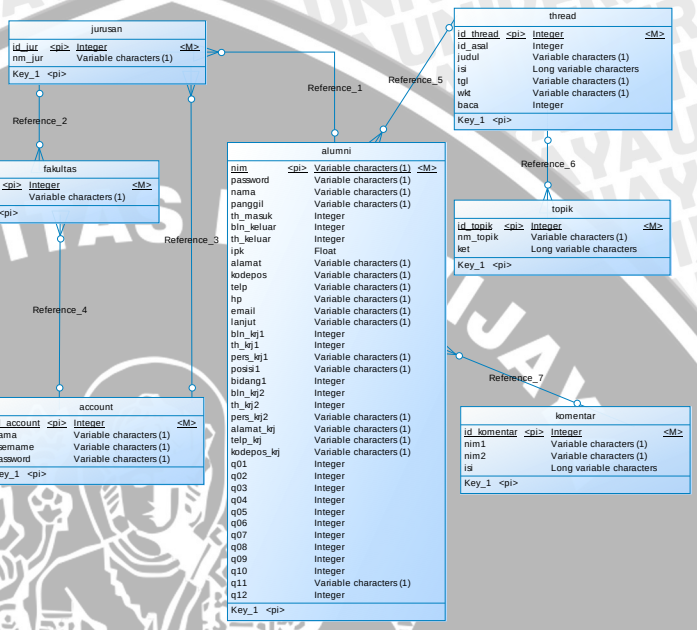
Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa tabel pada basis data `db_tracer`, hasil perancangan sesuai dengan tabel hasil implementasi.

Prosedur Uji :

1. Sebuah *window* Command Prompt dijalankan dari:
   `Start | Run... | Open: cmd.exe`
2. *Server* basis data MySQL dijalankan sebagai *service* dengan memberikan perintah:
   ```
   C:\Documents and
   Settings\Administrator>net start mysql
   ```
3. Memasuki SQL *Shell* dengan perintah berikut:
   ```
   C:\xampp\mysql\bin>mysql -u root -p
   db_tracer

   Enter password:
   ```
4. Tabel-tabel yang terdapat pada basis data db_tracer ditampilkan dengan menggunakan perintah SQL berikut:
   ```
   mysql>show tables;
   ```
5. Membuka *software* Sybase PowerDesigner 12.5 dengan cara berikut:
   `Start | All Programs | Sybase | PowerDesigner 12.5 | PowerDesigner`
6. Menggambarkan *Entity Relationalship Diagram*

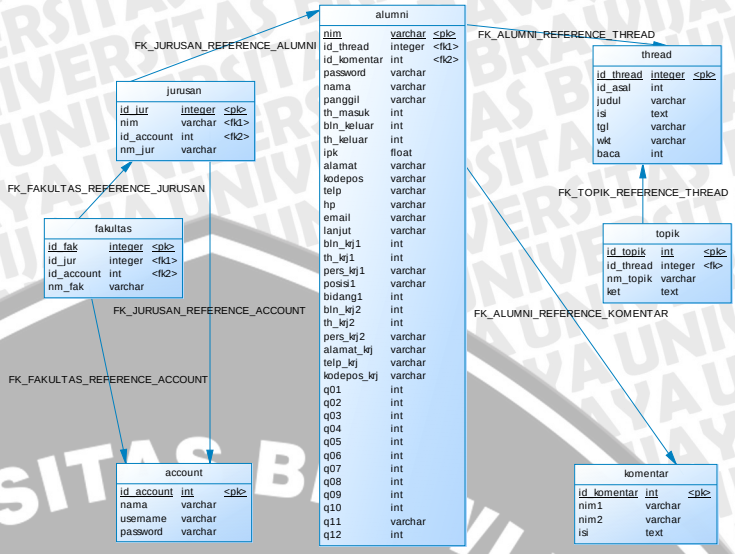
pada area kerja *Conceptual Data Model* (CDM).

7. Memeriksa diagram ER tersebut dengan cara menekan tombol *Check Model* pada *toolbar*. Hasil pemeriksaan ini disebut dengan CDM *Object* yang ditunjukkan dalam Gambar 6.2.



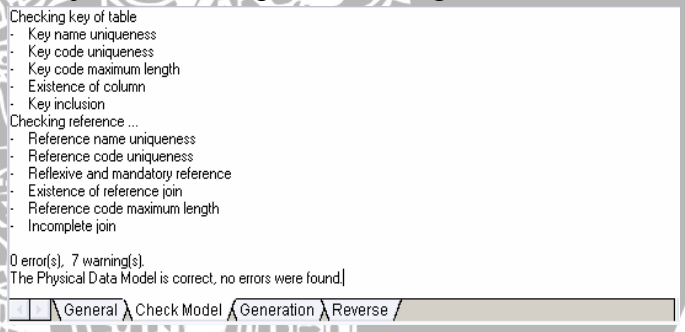
Gambar 6.2 *Conceptual Data Model Object*
Sumber: [Pengujian]

8. Untuk mengubah diagram ER dari CDM *Object* menjadi *Physical Data Model* (PDM) *Object*, dilakukan proses *generate* dengan menekan *link Generate Physical Data Model* pada *toolbar Tools*. PDM *Object* ditunjukkan dalam Gambar 6.3.



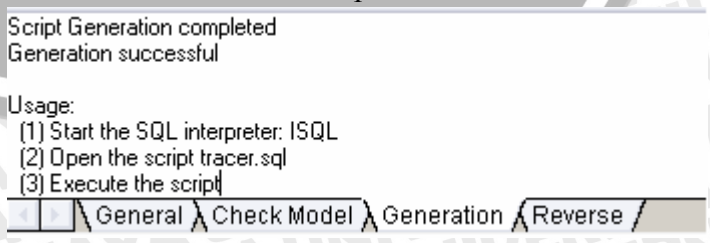
Gambar 6.3 *Physical Data Model*
Sumber: [Pengujian]

9. Setelah berubah menjadi PDM *Object*, dilakukan pemeriksaan kembali dengan cara menekan tombol *Check Model* pada *toolbar*. Hasil Pemeriksaan ditunjukan pada gambar 6.4.



Gambar 6.4 *Laporan Check Model*
Sumber: [Pengujian]

10. Setelah berubah menjadi PDM *Object*, dilakukan *generate* ke *database* MySQL dengan cara menekan *link Generate Database* pada *toolbar Tools*.



Gambar 6.5 *Laporan Check Model*
Sumber: [Pengujian]

Hasil Pengujian :

Tabel pada basis data `db_tracer` berhasil di *generate*, dan sesuai dengan tabel hasil implementasi. Hasil *generate* bisa dilihat dengan perintah `C:\xampp\mysql\bin>mysql -u root -p db_tracer` sesuai dengan prosedur uji, point 1 sampai dengan point 4.

6.1.2 Pengujian Koneksi Basis Data dan *Web Server*

Nama Kasus Uji : Kasus Uji Koneksi *Database*

Prosedur Uji : PC *Server* Aplikasi:

1. Sebuah *window* Command Prompt dijalankan dari:
   Start | Run... | Open: cmd.exe
2. *Server database* MySQL dijalankan sebagai *service* dengan memberikan perintah:
   C:\>net start mysql
3. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan dengan memberikan perintah:
   C:\>netstat –an

PC *Client*:

1. Membuka aplikasi *web* Penelusuran Data Alumni Universitas Brawijaya (*Tracer Study*) (http://172.17.8.34:8080/tracer).
2. Melakukan proses registrasi, *login* sebagai Alumni, dan *login* sebagai Administrator.
3. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan dengan memberikan perintah:
   C:\>netstat –an

PC *Server* Aplikasi:

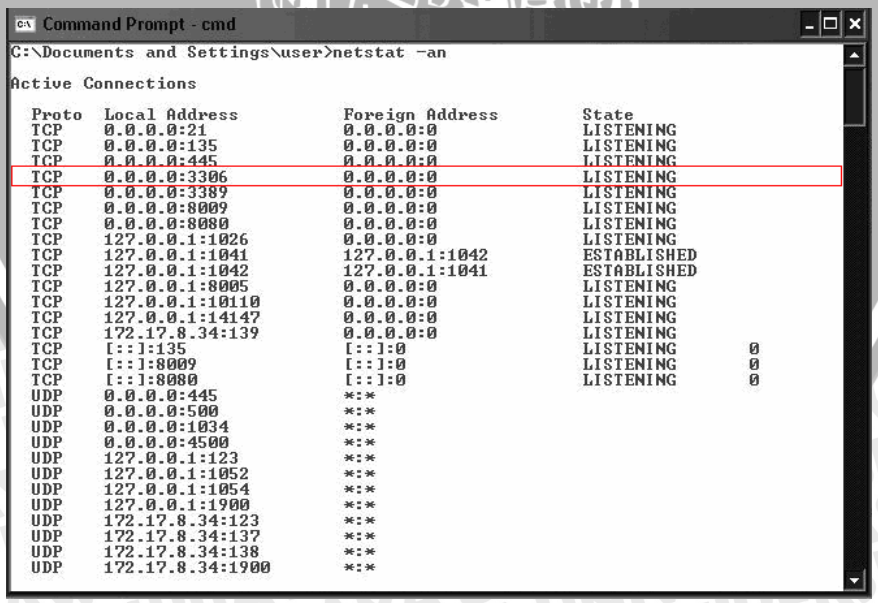
1. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan kembali dengan memberikan perintah:

   C:\>netstat –an

Hasil yang diharapkan : Komputer *client* dapat melakukan koneksi dengan *database* MySQL yang berada pada komputer *server*.

Hasil Pengujian : Komputer *client* dapat melakukan koneksi dengan *database* MySQL yang berada pada komputer *server*.

Hasil dari penggunaan perintah netstat –an pada komputer *server* aplikasi sebelum ada koneksi dengan komputer *client* ditunjukkan dalam Gambar 6.6. Perintah tersebut digunakan untuk menampilkan koneksi yang sedang aktif. Dari gambar tersebut terlihat bahwa *database server* MySQL (`mysqld-nt.exe`) memiliki kondisi (*state*) `LISTENING` pada alamat lokal 0.0.0.0:3306. Hal tersebut berarti bahwa *database server* MySQL telah siap untuk menerima sebuah koneksi *database* pada *port* TCP 3306.



Gambar 6.6 Koneksi yang sedang aktif pada komputer *server* sebelum aplikasi dijalankan pada komputer *client*
Sumber: [Pengujian]

6.1.3 Pengujian Waktu Akses *Query*

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan dalam melakukan *query* pada basis data `db_tracer`. Prosedur yang dijalankan dengan mengakses `query_tes.php`. File `query_tes.php` dibuat khusus untuk melakukan pengujian waktu akses query dengan menggunakan fungsi `microtime()` pada PHP.

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `account` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.7 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.1765 ms Query 2 = 0.1265 ms Query 3 = 0.124 ms Query 4 = 0.14 ms Query 5 = 0.1265 ms Query 6 = 0.125 ms Query 7 = 0.1285 ms Query 8 = 0.1655 ms Query 9 = 0.126 ms Query 10 = 0.1245 ms Rata - rata waktu query = 0.1363 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 1.444 ms Query 2 = 0.617 ms Query 3 = 0.347 ms Query 4 = 0.327 ms Query 5 = 0.323 ms Query 6 = 0.306 ms Query 7 = 0.327 ms Query 8 = 0.304 ms Query 9 = 0.319 ms Query 10 = 0.287 ms Rata - rata waktu query = 0.4601 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 0.722 ms Query 2 = 0.514 ms Query 3 = 0.5 ms Query 4 = 0.506 ms Query 5 = 0.506 ms Query 6 = 0.502 ms Query 7 = 0.506 ms Query 8 = 0.502 ms Query 9 = 0.502 ms Query 10 = 0.502 ms Rata - rata waktu query = 0.5262 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 1.792 ms Query 2 = 1.304 ms Query 3 = 1.344 ms Query 4 = 1.332 ms Query 5 = 1.268 ms Query 6 = 1.276 ms Query 7 = 1.3 ms Query 8 = 1.26 ms Query 9 = 1.3 ms Query 10 = 1.264 ms Rata - rata waktu query = 1.344 ms

Gambar 6.7 Hasil Pengujian Waktu Akses *Query* untuk Tabel `account`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `alumni` dengan jumlah data sebanyak 500 dan 1000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.8 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.1825 ms Query 2 = 0.1215 ms Query 3 = 0.119 ms Query 4 = 0.121 ms Query 5 = 0.12 ms Query 6 = 0.1195 ms Query 7 = 0.124 ms Query 8 = 0.1205 ms Query 9 = 0.1205 ms Query 10 = 0.12 ms Rata - rata waktu query = 0.12685 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 0.361 ms Query 2 = 0.256 ms Query 3 = 0.252 ms Query 4 = 0.253 ms Query 5 = 0.25 ms Query 6 = 0.255 ms Query 7 = 0.26 ms Query 8 = 0.253 ms Query 9 = 0.253 ms Query 10 = 0.252 ms Rata - rata waktu query = 0.2645 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 0.756 ms Query 2 = 0.512 ms Query 3 = 0.506 ms Query 4 = 0.506 ms Query 5 = 0.508 ms Query 6 = 0.508 ms Query 7 = 0.526 ms Query 8 = 0.51 ms Query 9 = 0.514 ms Query 10 = 0.508 ms Rata - rata waktu query = 0.5354 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 2.784 ms Query 2 = 1.588 ms Query 3 = 1.544 ms Query 4 = 1.52 ms Query 5 = 1.716 ms Query 6 = 1.864 ms Query 7 = 1.632 ms Query 8 = 1.308 ms Query 9 = 1.344 ms Query 10 = 1.28 ms Rata - rata waktu query = 1.658 ms

Gambar 6.8 Hasil Pengujian Waktu Akses *Query* untuk Tabel `alumni`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `fakultas` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry*

dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.9 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.1765 ms Query 2 = 0.128 ms Query 3 = 0.1245 ms Query 4 = 0.1235 ms Query 5 = 0.1215 ms Query 6 = 0.1235 ms Query 7 = 0.128 ms Query 8 = 0.1255 ms Query 9 = 0.125 ms Query 10 = 0.1245 ms Rata - rata waktu query = 0.13005 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 1.087 ms Query 2 = 0.332 ms Query 3 = 0.35 ms Query 4 = 0.293 ms Query 5 = 0.287 ms Query 6 = 0.334 ms Query 7 = 0.403 ms Query 8 = 0.308 ms Query 9 = 0.281 ms Query 10 = 0.264 ms Rata - rata waktu query = 0.3939 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 2.03 ms Query 2 = 0.732 ms Query 3 = 0.606 ms Query 4 = 0.686 ms Query 5 = 0.58 ms Query 6 = 0.584 ms Query 7 = 0.608 ms Query 8 = 0.536 ms Query 9 = 1.048 ms Query 10 = 1.914 ms Rata - rata waktu query = 0.9324 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 2.256 ms Query 2 = 1.396 ms Query 3 = 1.18 ms Query 4 = 1.164 ms Query 5 = 1.188 ms Query 6 = 1.16 ms Query 7 = 1.164 ms Query 8 = 1.164 ms Query 9 = 1.184 ms Query 10 = 1.164 ms Rata - rata waktu query = 1.302 ms

Gambar 6.9 Hasil Pengujian Waktu Akses *Query* untuk Tabel `fakultas`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `jurusan` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.10 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.1755 ms Query 2 = 0.125 ms Query 3 = 0.125 ms Query 4 = 0.123 ms Query 5 = 0.1245 ms Query 6 = 0.124 ms Query 7 = 0.1275 ms Query 8 = 0.1245 ms Query 9 = 0.1245 ms Query 10 = 0.125 ms Rata - rata waktu query = 0.12985 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 0.364 ms Query 2 = 0.263 ms Query 3 = 0.255 ms Query 4 = 0.358 ms Query 5 = 0.251 ms Query 6 = 0.25 ms Query 7 = 0.256 ms Query 8 = 0.249 ms Query 9 = 0.247 ms Query 10 = 0.251 ms Rata - rata waktu query = 0.2744 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 0.958 ms Query 2 = 0.66 ms Query 3 = 0.634 ms Query 4 = 0.608 ms Query 5 = 0.598 ms Query 6 = 0.59 ms Query 7 = 0.59 ms Query 8 = 0.592 ms Query 9 = 0.59 ms Query 10 = 0.594 ms Rata - rata waktu query = 0.6414 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 2.248 ms Query 2 = 1.628 ms Query 3 = 1.572 ms Query 4 = 1.56 ms Query 5 = 1.528 ms Query 6 = 1.532 ms Query 7 = 1.58 ms Query 8 = 1.56 ms Query 9 = 1.564 ms Query 10 = 1.544 ms Rata - rata waktu query = 1.6316 ms

Gambar 6.10 Hasil Pengujian Waktu Akses *Query* untuk Tabel `jurusan`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `komentar` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.11 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.427 ms Query 2 = 0.1705 ms Query 3 = 0.1535 ms Query 4 = 0.1545 ms Query 5 = 0.14 ms Query 6 = 0.182 ms Query 7 = 0.147 ms Query 8 = 0.128 ms Query 9 = 0.1265 ms Query 10 = 0.1255 ms Rata - rata waktu query = 0.17545 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 0.661 ms Query 2 = 0.361 ms Query 3 = 0.413 ms Query 4 = 0.348 ms Query 5 = 0.315 ms Query 6 = 0.31 ms Query 7 = 0.31 ms Query 8 = 0.306 ms Query 9 = 0.31 ms Query 10 = 0.308 ms Rata - rata waktu query = 0.3642 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 1.14 ms Query 2 = 0.804 ms Query 3 = 0.708 ms Query 4 = 0.664 ms Query 5 = 0.652 ms Query 6 = 0.686 ms Query 7 = 0.668 ms Query 8 = 0.702 ms Query 9 = 0.658 ms Query 10 = 0.668 ms Rata - rata waktu query = 0.735 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 1.784 ms Query 2 = 1.204 ms Query 3 = 1.164 ms Query 4 = 1.168 ms Query 5 = 3.06 ms Query 6 = 1.508 ms Query 7 = 1.32 ms Query 8 = 1.188 ms Query 9 = 1.172 ms Query 10 = 1.204 ms Rata - rata waktu query = 1.4772 ms

Gambar 6.11 Hasil Pengujian Waktu Akses *Query* untuk Tabel `komentar`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `thread` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.12 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.194 ms Query 2 = 0.126 ms Query 3 = 0.123 ms Query 4 = 0.1235 ms Query 5 = 0.1225 ms Query 6 = 0.1225 ms Query 7 = 0.1395 ms Query 8 = 0.134 ms Query 9 = 0.124 ms Query 10 = 0.1245 ms Rata - rata waktu query = 0.13335 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 0.408 ms Query 2 = 0.257 ms Query 3 = 0.254 ms Query 4 = 0.25 ms Query 5 = 0.252 ms Query 6 = 0.248 ms Query 7 = 0.251 ms Query 8 = 0.252 ms Query 9 = 0.253 ms Query 10 = 0.249 ms Rata - rata waktu query = 0.2674 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 0.966 ms Query 2 = 0.65 ms Query 3 = 0.634 ms Query 4 = 0.64 ms Query 5 = 0.63 ms Query 6 = 0.648 ms Query 7 = 0.626 ms Query 8 = 0.636 ms Query 9 = 0.648 ms Query 10 = 0.64 ms Rata - rata waktu query = 0.6718 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 4.368 ms Query 2 = 3.18 ms Query 3 = 1.744 ms Query 4 = 1.552 ms Query 5 = 1.476 ms Query 6 = 1.54 ms Query 7 = 1.448 ms Query 8 = 1.672 ms Query 9 = 1.976 ms Query 10 = 1.696 ms Rata - rata waktu query = 2.0652 ms

Gambar 6.12 Hasil Pengujian Waktu Akses *Query* untuk Tabel `thread`
Sumber: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `topik` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.13 berikut.

Loop = 10 Jumlah Data = 500 Query 1 = 0.289 ms Query 2 = 0.1605 ms Query 3 = 0.1605 ms Query 4 = 0.146 ms Query 5 = 0.144 ms Query 6 = 0.142 ms Query 7 = 0.1415 ms Query 8 = 0.128 ms Query 9 = 0.1295 ms Query 10 = 0.126 ms Rata - rata waktu query = 0.1567 ms	Loop = 10 Jumlah Data = 1000 Query 1 = 0.662 ms Query 2 = 0.328 ms Query 3 = 0.321 ms Query 4 = 0.321 ms Query 5 = 0.282 ms Query 6 = 0.29 ms Query 7 = 0.287 ms Query 8 = 0.293 ms Query 9 = 0.26 ms Query 10 = 0.256 ms Rata - rata waktu query = 0.33 ms	Loop = 10 Jumlah Data = 2000 Query 1 = 1.9 ms Query 2 = 0.938 ms Query 3 = 0.83 ms Query 4 = 0.82 ms Query 5 = 0.844 ms Query 6 = 0.83 ms Query 7 = 0.664 ms Query 8 = 0.674 ms Query 9 = 0.656 ms Query 10 = 0.642 ms Rata - rata waktu query = 0.8798 ms	Loop = 10 Jumlah Data = 4000 Query 1 = 1.944 ms Query 2 = 1.296 ms Query 3 = 1.308 ms Query 4 = 1.276 ms Query 5 = 1.252 ms Query 6 = 1.284 ms Query 7 = 1.252 ms Query 8 = 1.252 ms Query 9 = 1.296 ms Query 10 = 1.256 ms Rata - rata waktu query = 1.3416 ms

Gambar 6.13 Hasil Pengujian Waktu Akses *Query* untuk Tabel `topik`
Sumber: [Pengujian]

Hasil rata-rata pengujian waktu akses *query* terhadap basis data `db_tracer` dengan jumlah data sebanyak 500 dan 1000 data *entry* memberikan hasil yang cukup stabil. Data rata-rata pengujian terhadap basis data `db_tracer` ditunjukkan dalam Tabel 6.1.

Tabel 6.1 Pengujian Waktu Akses *Query*

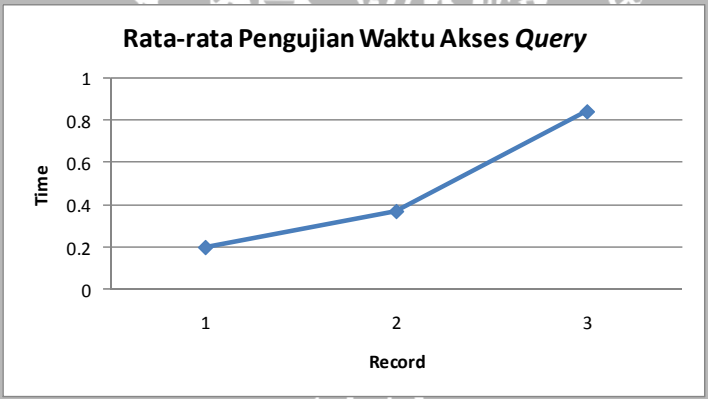
Nama Tabel	Waktu Akses dengan 500 Data *Entry* (ms)	Waktu Akses dengan 1000 Data *Entry* (ms)	Waktu Akses dengan 2000 Data *Entry* (ms)	Waktu Akses dengan 4000 Data *Entry* (ms)
account	0.1363	0.4601	0.5262	1.344
alumni	0.12685	0.2645	0.5354	1.658
fakultas	0.13005	0.3939	0.9324	1.302
jurusan	0.12985	0.2744	0.6414	1.6316
komentar	0.17545	0.3642	0.735	1.4772
thread	0.13335	0.2674	0.6718	2.0652
topik	0.1567	0.33	0.8798	1.3416

Sumber: [Pengujian]

Tabel 6.2 Rata-rata Pengujian Waktu Akses Query

	Δ_1	Δ_2	Δ_3
	0.3238	0.0661	0.8178
	0.13765	0.2709	1.1226
	0.26385	0.5385	0.3696
	0.14455	0.367	0.9902
	0.18875	0.3708	0.7422
	0.13405	0.4044	1.3934
	0.1733	0.5498	0.4618
Δ	**0.195135714**	**0.366785714**	**0.842514286**

Sumber: [Pengujian]



Gambar 6.14 Grafik Rata-rata Pengujian Waktu
Sumber: [Pengujian]

6.2 Pengujian Perangkat Lunak

Pengujian perangkat lunak sistem ini dibagi menjadi dua tahap, yaitu pengujian unit dan pengujian integrasi.

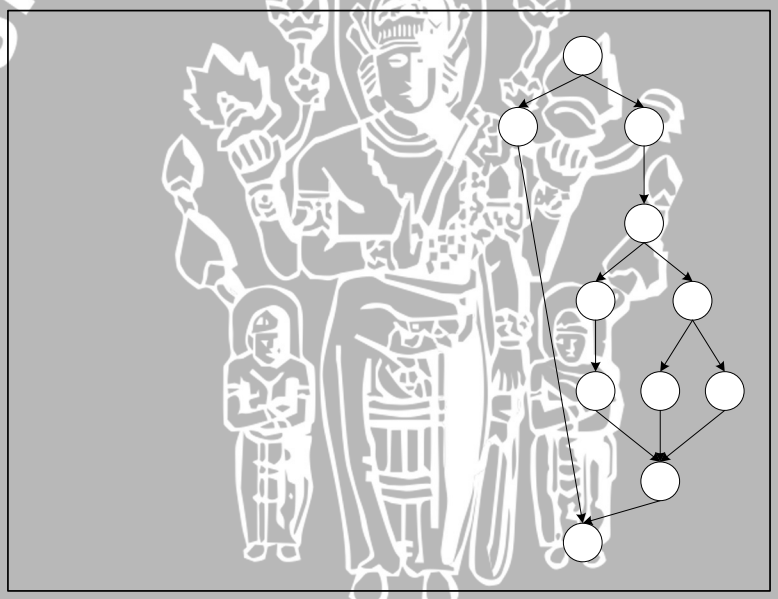
6.2.1 Pengujian Unit

Dalam sistem yang dibangun dengan pemrograman berorientasi objek, pengujian unit diterapkan untuk suatu metode (operasi) dari suatu *class*. Pada

pengujian unit ini, digunakan teknik pengujian *white box (white box testing)* dengan teknik *basis path testing*. Pada teknik *basis path testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph,* menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity),* menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Di dalam penulisan laporan skripsi ini, hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

6.2.1.1 Pengujian Unit untuk Operasi menambahAccount

Operasi menambahAccount merupakan salah satu metode yang terdapat dalam class MyAccount. Pada Gambar 6.15 memperlihatkan proses pemodelan dalam *flow graph* pada operasi menambahAccount pada class MyAccount.



Gambar 6.15 *Flowgraph* pada Operasi menambahAccount
Sumber: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.15 yang telah dilakukan terhadap operasi menambahAccount, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

```
function Simpan()
        {
        $user=$_POST[ user ];
        $str = "select username from account where u
        $qry = mysql_query ($str) or die (mysql_error(
```

$$V(G) = E - N + 2$$
$$= 13 - 11 + 2$$
$$= 4$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 4 ditentukan dua basis set dari jalur independen yaitu:

Jalur 1 : 1-2-11

Jalur 2 : 1-3-4-5-7-10-11

Jalur 3 : 1-3-4-6-8-10-11

Jalur 4 : 1-3-4-6-9-10-11

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.3.

Tabel 6.3 *Testcase* untuk Pengujian Unit Operasi menambahAccount

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Data yang dimasukkan sudah ada dalam database	Dikembalikan ke *class* MyAccount	Dikembalikan ke *class* MyAccount
2	Memanggil Admin.memasukanAccount (nama, user, pass)	Sistem Menampilkan Kembalian Berupa String	Sistem Menampilkan Kembalian Berupa String
3	Memassukkan Account untuk Fakultas (nama, user, pass, id_fakultas)	Sistem menampilkan kembalian berupa string	Sistem menampilkan kembalian berupa string
4	Memassukkan Account untuk Jurusan (nama, user, pass, id_jurusan)	Sistem menampilkan kembalian berupa string	Sistem menampilkan kembalian berupa string

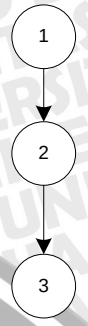
Sumber: [Pengujian]

6.2.1.2 Pengujian Unit untuk Operasi mengisiRegistrasi

Operasi mengisiRegistrasi merupakan salah satu metode yang terdapat dalam *class* MyAlumni. Pada Gambar 6.16 memperlihatkan proses pemodelan dalam *flow graph* pada operasi mengisiRegistrasi pada *class* MyAlumni.

```
function Simpan()
{
$str = "insert into alumni set nim='$_POST[nim]', nama='$_POST[nama1]',
panggil='$_POST[nama2]', tempat='$_POST[tempat]', tgl='$_POST[tgl]',
bln='$_POST[bln]',  thn='$_POST[thn]', password='$_POST[pass]',
id_jur='$_POST[id_jur]', th_masuk='$_POST[th1]',
bln_keluar='$_POST[bln_keluar]', th_keluar='$_POST[th_keluar]',
ipk=$_POST[ipk], alamat='$_POST[alamat]', kodepos='$_POST[pos]',
telp='$_POST[telp]', hp='$_POST[hp]', email='$_POST[email]',
bln_krj1='$_POST[bln_krj1]',  th_krj1=$_POST[th_krj1],
pers_krj1='$_POST[pers_krj1]', posisi1='$_POST[posisi1]',
bidang1='$_POST[bidang1]', q06=$_POST[q6], bln_krj2='$_POST[bln_krj2]',
th_krj2=$_POST[th_krj2], pers_krj2='$_POST[pers_krj2]',
q11='$_POST[q11]', q04='$_POST[q04]', alamat_krj='$_POST[alamat_krj]',
telp_krj='$_POST[telp_krj]', kodepos_krj='$_POST[kodepos_krj]'";

$qry = mysql_query ($str) or die (mysql_error());

echo "Data alumni sudah tersimpan...";
}
```

Edge (E) = 2
Node (N) = 3



Gambar 6.16 *Flowgraph* pada Operasi mengisiRegistrasi
Sumber: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.16 yang telah dilakukan terhadap operasi mengisiRegistrasi, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

$$V(G) = E - N + 2$$
$$= 2 - 3 + 2$$
$$= 1$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan dua basis set dari jalur independen yaitu:

Jalur : 1-2-3

Penentuan kasus uji *(test case)* untuk jalur tersebut dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.4.

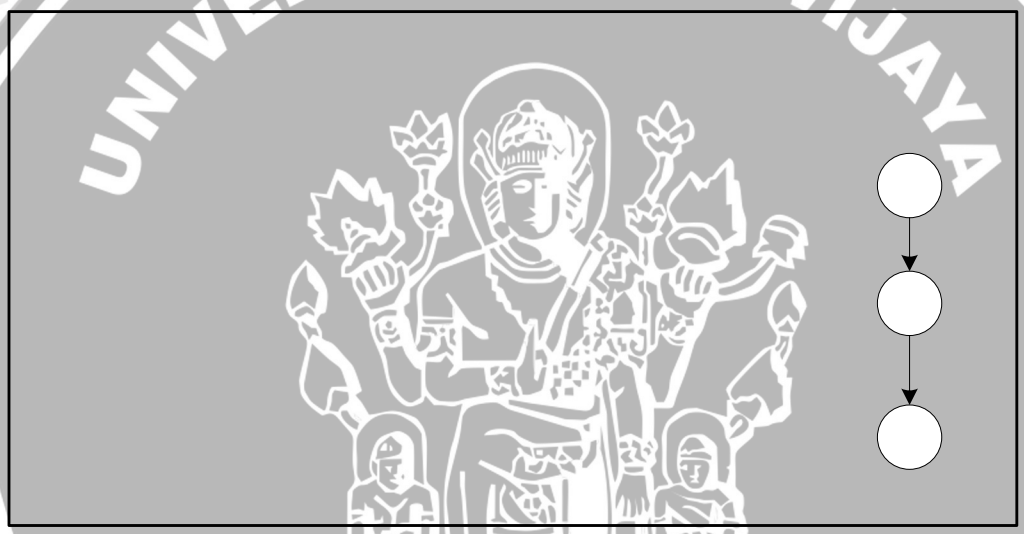
Tabel 6.4 *Testcase* untuk Pengujian Unit Operasi mengisiRegistrasi

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Memasukkan data registrasi ke dalam database (nama, nim, tgl_lahir, th_masuk, dll)	Sistem Menampilkan Kembalian Berupa String	Sistem Menampilkan Kembalian Berupa String

Sumber: [Pengujian]

6.2.1.3 Pengujian Unit untuk Operasi menambahFakultas

Operasi menambahFakultas merupakan salah satu metode yang terdapat dalam class MyFakultas. Pada Gambar 6.17 memperlihatkan proses pemodelan dalam *flow graph* pada operasi menambahFakultas pada class MyFakultas.



Gambar 6.17 *Flowgraph* untuk Operasi menambahFakultas
Sumber: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.17 yang telah dilakukan terhadap operasi menambahFakultas, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2

= 2 – 3 + 2

= 1

{

```
$nama=$_POST['nama'];
$str = "insert into fakultas
$qry = mysql_query ($str)
echo "Fakultas sudah ters
```

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan dua basis set dari jalur independen yaitu:

Jalur : 1-2-3

Penentuan kasus uji *(test case)* untuk jalur tersebut dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.5.

Tabel 6.5 *Testcase* untuk Pengujian Unit Operasi menambahFakultas

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Memasukkan nama Fakultas ke dalam database	Sistem Menampilkan Kembalian Berupa String	Sistem Menampilkan Kembalian Berupa String

Sumber: [Pengujian]

6.2.1.4 Pengujian Unit untuk Operasi menambahJurusan

Operasi menambahJurusan merupakan salah satu metode yang terdapat dalam class MyJurusan. Pada Gambar 6.18 memperlihatkan proses pemodelan dalam *flow graph* pada operasi menambahJurusan pada class MyJurusan.

Gambar 6.18 *Flowgraph* untuk Operasi menambahJurusan
Sumber: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.18 yang telah dilakukan terhadap operasi menambahJurusan, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

$$V(G) = E - N + 2$$
$$= 2 - 3 + 2$$
$$= 1$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan dua basis set dari jalur independen yaitu:

Jalur : 1-2-3

Penentuan kasus uji *(test case)* untuk jalur tersebut dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.6.

Tabel 6.6 *Testcase* untuk Pengujian Unit Operasi menambahJurusan

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Memasukkan data Jurusan ke dalam database (nama, id_fakultas)	Sistem Menampilkan Kembalian Berupa String	Sistem Menampilkan Kembalian Berupa String

Sumber: Pengujian

6.2.2 Pengujian Integrasi

Pengujian Integrasi diterapkan pada kebutuhan sistem yang mengintegrasikan fungsionalitas dari beberapa class untuk memenuhi suatu kebutuhan sistem. Pada pengujian Integrasi yang dijadikan objek uji adalah *Use-case* yang merupakan hasil dari analisis kebutuhan sistem.

Pengujian Integrasi didahului oleh pengujian pada unit-unit yang akan diintegrasikan. Pengujian integrasi ini menggunakan metode pengujian *Black box Testing*, karena pengujian ini lebih ditekankan untuk menemukan kesesuaian antara kinerja sistem dengan *Use-case*.

Penulisan Laporan Skripsi ini hanya dicantumkan hasil pengujian untuk *Use-case* mengisi kuisioner saja.

6.2.2.1 Pengujian Integrasi untuk Usecase Mengisi Kuisioner

Dalam memenuhi kebutuhan yang terdapat pada *Use-case* Mengisi Kuisioner diintegrasikan dua *class* yaitu *class* MyAlumni dan *class* MyQuiz. Metode-metode yang dilibat untuk menjawab kebutuhan dari *Use-case* Mengisi Kuisioner adalah mengisiRegistrasi dan mengisiKuisioner.

Tujuan pengujian integrasi ini adalah menguji apakah data yang dimasukan dapat tersimpan dengan baik pada basis data.

Kasus uji untuk pengujian *Use-case* Mengisi Kuisioner ditampilkan pada tabel dibawah ini.

Tabel 6.7 Kasus Uji untuk pengujian integrasi pada *Use-case* Mengisi Kuisioner

Kasus Uji	**:**	**Menjalankan prosedur untuk Mengisi Kuisioner melalui aplikasi Web Penelusuran Data Alumni Universitas Brawijaya**	
Prosedur	**:**	**1**	**Aktor melakukan Registrasi Alumni**
		2	**Aktor memasukkan data pribadi dan karier pada form registrasi**
		3	**Aktor login sebagai Alumni**
		4	**Aktor memilih submenu Kuisioner pada Menu Utama**
		5	**Aktor memasukkan data jawaban pada form kuisioner**
Hasil yang diharapkan	**:**	**Sistem dapat menyimpan Data Kuisioner yang telah dibuat ke dalam basis data dengan baik**	

Sumber : [Pengujian]

Hasil yang diperoleh dari hasil pengujian melalui kasus uji pada tabel diatas ditampilkan pada tabel dibawah ini.

Tabel 6.8 Hasil pengujian ntegrasi pada *Use-case* Mengisi Kuisioner

Hasil yang diharapkan	**:**	**Sistem dapat menyimpan Data Kuisioner yang telah dibuat kedalam basisdata dengan baik**
Hasil yang didapatkan	**:**	**Sistem dapat menyimpan Data Profil kedalam basis data**

		PROFIL LENGKAP ALUMNI Nama Lengkap : Fedrian Nuril Nama Panggilan : Fedi Tempat, Tanggal Lahir : Jakarta, 1 Juli 1982 Fakultas : teknik Jurusan : teknik mesin Tahun Masuk : 2000 Bulan, Tahun Keluar : Juni, 2005 IPK : 3.45 Alamat : Jl. Pelangi 28 Jakarta Kodepos : 12345 Telepon : 898989 Handphone : 08123456789 Email : fedi@yahoo.com **Sistem dapat menyimpan Data Kuisioner kedalam basis data** **Kuisioner** Data Kuisioner Sudah Tersimpan...

Sumber : [Pengujian]

Dari hasil pengujian integrasi pada *Use-case* Mengisi Kuisioner diketahui sistem dapat menyimpan data kuisioner yang telah dibuat kedalam basis data.

BAB VII

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil Perancangan, Implementasi, dan Pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut :

1. Hasil pengujian *table* menunjukan bahwa tabel basis data hasil implementasi pada aplikasi ini sesuai dengan tabel pada basis data hasil perancangan.
2. Hasil pengujian unit dan pengujian integrasi menunjukan bahwa sistem dapat melakukan proses pengolahan dan penyimpanan data Alumni, Fakultas, dan Jurusan sesuai dengan hasil perancangan.
3. Hasil pengujian waktu akses *query* dengan data entry 500, 1000, 2000, dan 4000 menunjukkan peningkatan linier dengan rata-rata waktu akses Δ_1 = 0.195135714 ms, Δ_2 = 0.366785714 ms, dan Δ_3 = 0.842514286 ms.
4. Aplikasi *Web* Penelusuran Data Alumni Universitas Brawijaya (*Tracer Study*) ini dapat mengumpulkan data-data alumni UB dan membuat *database* alumni sehingga akan mempermudah dalam pengelolaan dan pengembangan UB di masa yang akan datang.
5. Hasil dari seluruh rangkaian pengujian, menunjukan bahwa sistem yang dikembangkan adalah benar sesuai dengan perancangan.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut dari Aplikasi *Web* Penelusuran Data Alumni Universitas Brawijaya (*Tracer Study*) ini adalah :

1. Aplikasi *Web* Penelusuran Data Alumni Unversitas Brawijaya (*Tracer Study*) dapat dikembangkan menjadi salah satu *web* yang menjadi sarana komunikasi antar Alumni UB sehingga membentuk jejaring sosial Universitas Brawijaya.

DAFTAR PUSTAKA

[ANO-01] http://www.brawijaya.ac.id

[ANO-02] http://www.total.or.id/info

[DWI-05] Dwi Prasetyo, Didik. 2005. *Solusi Menjadi Web Master melalui Manajemen Web dengan PHP*. Elex Media Komputindo. Jakarta.

[HAR-06] Harrismare, 2006. *Pemodelan Database*
Akses dari :
http://dosen.amikom.ac.id/downloads/materi/DatabaseModeling(ERD-ERM).pdf

[IRA-06] Irawan, Ivan. 2006. *PHP? Siapa Takut!*
Akses dari :
http://ilmukomputer.com/2006/09/13/php-siapa-takut/

[IRM-03] Irmansyah, Faried, 2003. *Pengantar Database*
Akses dari :
http://ilmukomputer.com:81/umum/faried-database.php.

[ISM-05] Ismunandar, Sunarto. 2005. *Strategic Planning Capacity (Tracer Study on IESD Strategic Planning)*. Universitas Brawijaya. Malang.

[JUH-06] Juhana, Tutun. 2006. *World Wide Web Security*. STEI ITB.
Akses dari:
http://zulidamel.wordpress.com/2007/09/17/perangkat-keras-jaringan-komputer/

[KUR-06] Kurniawan, Erick. 2006. *OOP PHP5*.
Akses dari :
http://www.ukdw.ac.id/kuliah/si/erickblog/PemrogramanDasarPHP5_ECB0/ObjectOrientedProgrammingPHP5.pdf

[MOE-08] Moeljadi. 2008. *Tracer Study di Perguruan Tinggi*. Materi Pelatihan Unistaff 2004. Universitas Brawijaya. Malang.

[NUG-05] Nugroho, Bunafit. 2005. *Database Relasional dengan MySQL* . Andi Offset.Yogyakarta.

[NUG-08] Nugroho, Bunafit. 2008. *Latihan Membuat Aplikasi Web PHP dan MySQL dengan Dreamweaver MX (6, 7, 2004) dan 8*. Gava Media. Yogyakarta.

[PRA-04] Prasetyo, Didik Dwi. 2004. *Pemrograman Berbasis Web Menggunakan PHP5*. Elex Media Komputindo. Jakarta.

[RAH-09] Raharjo, Willy Sudiarto. 2009. *Pemodelan Sistem Perangkat Lunak UML: Use Case Diagram*.
Akses dari:
http://www.ukdw.ac.id/kuliah/si/uml_use_case.pdf

[RAH-09] Raharjo, Willy Sudiarto. 2009. *Pemodelan Sistem Perangkat Lunak: OOP dan Class Diagram*.
Akses dari:
http://lecturer.ukdw.ac.id/willysr/pspl-ti/uml_class2.pdf

[SAL-07] Salman, 2007, *Apache Web Server*
Akses dari :
http://www.ilmukomputer.com/wp-content/uploads/2007/03/salman-apachewebserver/

[SUH-04] Suharyanto, A. dan Tolle, H. 2004. *Tracer Study Fakultas Teknik Universitas Brawijaya*. Fakultas Teknik Universitas Brawijaya. Malang.

[SUY-04] Suyanto, Asep Herman. 2004. *Basis Data Dan DBMS.*
Akses dari :
http://www.asep-hs.web.ugm.ac.id/basisdatadandbms.pdf

[WAH-05] Wahyono, Teguh. 2005. *Pemrograman Web Dinamis dengan PHP*. Elex Media Komputindo. Jakarta.