MODUL PRAKTIKUM REKAYASA PERANGKAT LUNAK

Oleh:
Allin Junikhah, M.T.
Nur Fitriyah Ayu Tunjung Sari, M.Cs.
Tri Mukti Lestari, M.Kom.

Jurusan Teknik Informatika Fakultas Sains dan Teknologi
UIN Maulana Malik Ibrahim Malang
2022/2023

MODUL 1

PENGANTAR RPL

1.1 Bahasan dan Tujuan

1.1.1 Bahasan

Membahas tentang pengertian perangkat lunak dan rekayasa perangkat lunak secara umum.

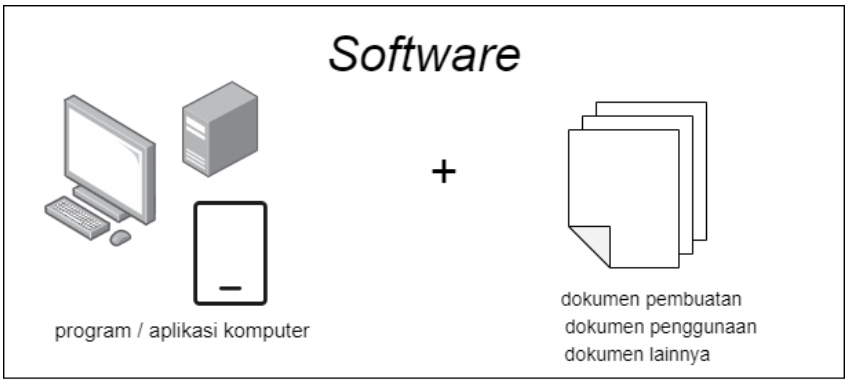
1.1.2 Tujuan

1. Mahasiswa memahami pengertian perangkat lunak secara umum dan peranannya.
2. Mahasiswa memahami pengertian rekayasa perangkat lunak dan prosesnya secara umum.

1.2 Dasar Teori

1.2.1 Perangkat Lunak

Perangkat lunak/*software* adalah program komputer beserta dengan berbagai dokumentasi terkait seperti persyaratan/*requirement*, model desain dan manual pengguna (*user guide*). Definisi resmi yang mengacu pada standard dari IEEE dan ISO yaitu *Software is computer programs, procedures, and possibly associated documentation and data pertaining to the operation of computer system*.



Karakteristik Perangkat Lunak:

a. Perangkat lunak dibangun dengan proses *software engineering* bukan diproduksi secara manufaktur atau pabrikan.
b. Perangkat lunak tidak pernah usang atau *wear out* karena kecacatan pada perangkat lunak dapat diperbaiki.
c. Perangkat lunak akan terus diperbaiki seiring dengan bertambahnya kebutuhan.

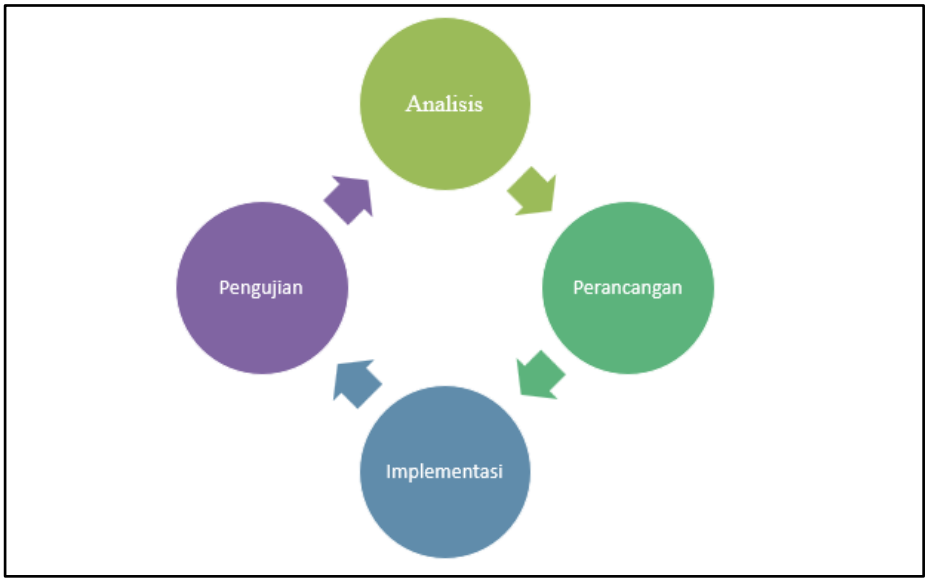
1.2.2 Rekayasa Perangkat Lunak

Sedangkan Rekayasa Perangkat Lunak atau *Software Engineering* adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal *requirement capturing* (analisa kebutuhan pengguna), *specification* (menentukan spesifikasi dari kebutuhan pengguna), desain, *coding*, *testing* sampai pemeliharaan sistem setelah digunakan. (Romi Satria Wahono berdasar pendapat Ian Sommerville). Rekayasa perangkat lunak lebih fokus kepada praktek pengembangan perangkat lunak yang memenuhi kriteria:

a. *Maintainability*, dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring perkembangan teknologi dan lingkungan.
b. *Dependability* dan *robust*, dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi.
c. *Usability*, kemampuan untuk digunakan sesuai dengan kebutuhan.
d. *Efficient*, efisien dari segi sumber daya dan penggunaan.

Berdasarkan kriteria di atas maka dapat disimpulkan bahwa perangkat lunak yang baik adalah perangkat lunak yang fokus pada kebutuhan pengguna atau pelanggan.

Tahapan Umum Rekayasa Perangkat Lunak



Berdasarkan diagram di atas dapat disimpulkan bahwa proses pengembangan perangkat lunak dapat dikaregorikan ke dalam tiga fase umum yaitu:

a. Fase pendefinisian yang fokus pada "***What, Who, When, Where"***.
b. Fase pengembangan yang fokus dengan ***"How"***.
c. Fase pendukung atau *support phase* yang fokus pada perubahan pada perbaikan atau *error*.

1.3 TUGAS PRAKTIKUM

1. Bentuk kelompok dimana satu kelompok terdiri dari 2 orang.
2. Lakukan curah pendapat (*brain storming*) untuk sebanyak mungkin memunculkan ide permasalahan/studi kasus yang akan dibuatkan solusi perangkat lunaknya.
3. Menyusun daftar sekaligus prioritas dari ide yang berpotensi untuk diangkat menjadi studi kasus.
4. Merumuskan permasalahan dari masing-masing ide tersebut sekaligus solusi yang dari perangkat lunak yang akan dibuat dalam satu semester.
5. Buat laporan untuk ide studi kasus terpilih dengan susunan laporan sebagai berikut:
    a. Judul perangkat lunak.
    b. Permasalahan.
    c. Solusi.
    d. Perangkat lunak yang akan dibuat.
    e. Target market atau pengguna.

MODUL 2

***System Development Life Cycle* (SDLC)**

1.1 Bahasan dan Tujuan

1.1.1 Bahasan

Membahas tentang pengertian *System Development Life Cycle* (SDLC)

1.1.2 Tujuan

1. Mahasiswa memahami ragam metodologi pengembagan perangkat lunak
2. Mahasiswa mampu membuat pengembangan perangkat lunak

1.2 Dasar Teori

Pengembangan sistem informasi berbasis komputer merupakan suatu kegiatan yang sangat kompleks, memerlukan biaya dan sumber daya yang besar, dan merupakan kegiatan yang sangat penting bagi organisasi (Hoffer, 2011). Kompleksitas ini muncul disebabkan oleh beberapa hal:

1. Sebuah sistem yang akan dikembangkan melibatkan pemangku kepentingan (stakeholder) yang beragam dengan keinginan, kebutuhan, dan agenda yang berbeda-beda.

2. Rumitnya struktur proses bisnis dalam organisasi di mana sistem informasi (berbasis komputer) tersebut akan dikembangkan. Sebagaimana diketahui bahwa sistem informasi berbasis komputer dikembangkan merujuk pada sistem bisnis (proses bisnis) organisasi.

3. Rumit dan lamanya proses pengembangan sistem informasi itu sendiri, terutama pada sistem-sistem yang berskala besar. Dalam situasi yang kompleks ini, suatu proyek pengembangan sistem informasi harus dijalankan dengan suatu prosedur kerja yang terstruktur dan terukur agar efisien dalam mencapai tujuan yang telah ditetapkan.

A. SIKLUS HIDUP PENGEMBANGAN SISTEM

Pengembangan sistem informasi yang berbasis komputer merupakan suatu kegiatan yang cukup kompleks yang membutuhkan banyak sumber daya dan dapat memakan waktu yang lama untuk menyelesaikannya. Proses pengembangan sistem melewati beberapa tahapan (fase), dimulai dari sistem itu direncanakan sampai dengan sistem tersebut diterapkan, dioperasikan, dan dipelihara. Bila operasi sistem yang sudah dikembangkan masih timbul permasalahan-permasalahan yang kritis serta tidak dapat diatasi dalam tahap pemeliharaan sistem, maka sistem tersebut perlu ditinjau kembali untuk dikembangkan dengan mengimplementasikan kembali tahap awal, yaitu tahap perencanaan sistem. Siklus ini disebut dengan siklus hidup suatu sistem (*system life cycle*) yang biasa disebut dengan istilah Siklus Hidup Pengembangan Sistem (System Development Life Cycle/SDLC). Daur atau siklus hidup dari pengembangan sistem merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan-tahapan utama dan langkah-langkah di dalam tahapan tersebut dalam proses pengembangannya.

Tahapan SDLC Dalam banyak hal, membangun (mengembangkan) sistem informasi mirip dengan membangun rumah. Pertama, pemilik menjelaskan visi rumah kepada pengembang. Kedua, ide ini ditransformasikan menjadi sketsa dan gambar yang diperlihatkan kepada pemilik (seringkali melalui beberapa gambar) sampai pemilik setuju

bahwa gambar-gambar tersebut telah menggambarkan apa yang pemilik inginkan. Ketiga, serangkaian cetak biru (blueprint) terperinci dikembangkan yang menyajikan lebih banyak informasi spesifik tentang rumah (misalnya tata letak kamar, penempatan perlengkapan pipa dan jaringan listrik, dan sebagainya). Akhirnya, rumah dibangun mengikuti cetak biru, dan seringkali dengan beberapa perubahan yang dibuat oleh pemilik ketika rumah itu didirikan. Ide dari system life cycle adalah sederhana dan logis. Tiap-tiap bagian dari pengembangan sistem dibagi menjadi beberapa tahapan kerja dan setiap tahapan mempunyai karakteristik tersendiri. Tahapan utama siklus hidup pengembangan sistem terdiri dari tahapan perencanaan sistem (system planning), analisis sistem (system analysis), desain sistem (system design), implementasi sistem (system implementation), dan pemeliharaan sistem (system maintenance). Tahapan-tahapan seperti ini sebenarnya merupakan tahapan di dalam pengembangan sistem teknik (engineering system), misalnya teknik pengembangan konstruksi gedung, mesin, dan teknik pengembangan perangkat lunak (software engineering). Istilah software engineering merupakan proses pengembangan perangkat lunak yang merupakan sub sistem dari pengembangan sistem informasi. Dengan demikian terdapat kesamaan metode-metode yang dibahas pada pengembangan sistem informasi dengan metode-metode pada pengembangan perangkat lunak (software). Ada banyak varian tahapan dari SDLC sejak tercatat pertama kali digunakan di Inggris pada akhir dekade 1960-an (Avison & Fitzgerald, 2006). Penggunaan tersebut diusulkan oleh National Computing Centre (NCC) yang disponsori oleh pemerintah Inggris. Usulan ini sebagai bentuk standardisasi metodologi bagi pengembangan sistem informasi di kalangan pemerintah Inggris maupun pihak swasta. Pada era tahun 1970-an, beberapa siklus pengembangan sistem dikemukakan oleh beberapa ahli di bidang pengembangan sistem (Jogiyanto, 2008). Pada umumnya tahapan-tahapan utama proses pengembangan sistem yang dikemukakan adalah sama.

Dari beberapa siklus pengembangan sistem yang dikemukakan oleh para ahli, terdapat tiga siklus utama yaitu analisis sistem, desain sistem, dan implementasi sistem. Beberapa penulis memasukkan proses kebijakan dan perencanaan sistem dalam tahapan pengembangan sistem. Tahapan ini sebenarnya adalah sebagai awal terjadinya proyek sistem (initiation of system project). Beberapa penulis juga membagi tahapan desain sistem menjadi desain sistem secara umum atau desain sistem secara konsep atau desain sistem secara logika dan desain sistem secara rinci atau desain sistem secara fisik. Tahap perawatan sistem (system maintenance) sebenarnya juga merupakan tahapan setelah pengembangan sistem selesai dilakukan dan sistem dioperasikan. Beberapa penulis menyebutnya sebagai tahap manajemen sistem, karena yang melakukan proses ini sudah bukan analis sistem, akan tetapi manajemen.

Uraian berikut ini menyajikan gambaran secara umum proses yang berlangsung pada setiap tahapan pada SDLC (Sarosa, 2017):

a. Perencanaan dan Studi Kelayakan Proyek

Sistem Studi kelayakan pada intinya mencoba meninjau apakah kebutuhan akan pengembangan sistem informasi baru (baik sistem baru sama sekali maupun pengganti sistem yang lama) layak secara ekonomis maupun dari sisi kelayakan lainnya. Kriteria kelayakan yang perlu diperhatikan dalam suatu studi kelayakan adalah adalah sebagai berikut (Romney, 2012).

1) Kelayakan secara hukum, yaitu tidak melanggar aturan maupun

Undang- undang yang berlaku.

2) Kelayakan teknis, yaitu tersedia teknologi untuk membangun sistem serta

tersedia sumber daya manusia (tenaga ahli) dalam membangun sistem.

3) Kelayakan ekonomi, yaitu biaya membangun sistem lebih kecil daripada

manfaat yang diperoleh.

4) Kelayakan organisasi dan sosial, yaitu sistem baru dapat diterima oleh para

anggota organisasi. Laporan hasil studi kelayakan menjadi dasar bagi manajemen untuk memutuskan apakah akan meneruskan upaya pengembangan sistem baru atau tidak. Jika keputusannya adalah melanjutkan berarti akan masuk ke tahap berikutnya, yaitu investigasi sistem.

b. Penyelidikan dan Penelitian Sistem

Pada tahap ini dilakukan penelitian dalam rangka pencarian fakta yang lebih terperinci dengan cara menelusuri secara lebih mendetail sistem seperti apa yang dibutuhkan. Penelitian dilakukan terhadap hal-hal, yaitu:

1) permasalahan pada sistem yang ada saat ini;

2) kebutuhan fungsional dari sistem yang sedang berjalan saat ini, apakah telah terpenuhi atau tidak;

3) kebutuhan sistem baru yang mungkin muncul seiring dengan perkembangan waktu, perubahan situasi dan lingkungan bisnis, maupun perkembangan teknologi;

4) kendala sistem dan batasan-batasan yang tidak boleh dilampaui;

5) jenis dan volume data yang akan diolah dan disimpan;

6) pengecualian-pengecualian terhadap kondisi normal yang mungkin akan dihadapi sistem di masa mendatang.

Pencarian fakta dilakukan dengan menggunakan alat pengumpulan data berupa berikut ini.

1) Pengamatan terhadap cara kerja dan perilaku pengguna sistem yang ada. Hasil pengamatan dapat memberikan pemahaman akan masalah, kondisi kerja, metode kerja, dan kemampatan.
2) Wawancara terhadap individu maupun kelompok pengguna. Wawancara merupakan kesempatan baik untuk bertemu dengan pengguna dari berbagai tingkatan organisasi, mendengarkan apa kebutuhan dan keluhan mereka. Wawancara juga dapat digunakan untuk mengkomunikasikan perubahan yang akan terjadi dan mengurangi resistensi pengguna.
3) Kuesioner untuk mendapatkan informasi dari responden dalam jumlah besar dan secara geografis tersebar di berbagai lokasi yang berbeda.
4) Studi Pustaka dengan cara menelaah dokumen dan catatan-catatan yang

ada untuk mendapatkan pemahaman sistem dan menemukan masalah yang terjadi. Analis sistem harus sadar bahwa terkadang dokumen dan catatan yang ditemukan sudah terlalu tua dan tidak lagi mencerminkan kondisi yang ada saat ini.

Beberapa penulis ahli di bidang pengembangan sistem mengkategorikan kegiatan penyelidikan dan penelitian sistem menjadi bagian dari kegiatan analisis sistem.

c. Analisis Sistem

Data yang diperoleh dari kegiatan investigasi sistem dianalisis untuk menentukan;

1) Domain informasi, yang berisi tiga hal yaitu:

a) muatan dan hubungan informasi

b) aliran informasi

c) struktur informasi

2) Fungsi-fungsi yang akan dilakukan oleh sistem baru

3) Tingkah laku sistem

4) Model-model yang menggambarkan informasi, fungsi, dan tingkah laku Dalam tahapan analisis sistem, pengembang mencoba memahami sistem lama, mengapa dan bagaimana sistem tersebut dibuat, dan bagaimana sistem lama dapat diperbaiki atau dikembangkan.

d. **Perancangan (Desain)**

Sistem Tujuan dari fase analisis adalah untuk mengetahui kebutuhan bisnis sistem, sedangkan tujuan dari fase desain adalah untuk memutuskan bagaimana membangunnya. Selama bagian awal desain, tim proyek mengubah kebutuhan bisnis sistem menjadi kebutuhan sistem yang menggambarkan detail teknis untuk membangun sistem. Tidak seperti kebutuhan bisnis, yang dikomunikasikan melalui penggunaan model logis dan model data, kebutuhan sistem dikomunikasikan melalui kumpulan dokumen desain dan proses fisik serta model data. Secara bersama-sama, dokumen desain dan model fisik membentuk cetak biru untuk sistem baru yang dikembangkan (Dennis, 2012). Analis bisnis sering mengalihkan perhatian mereka ke desain bisnis sistem pada tahap proyek ini, sementara analis sistem fokus pada elemen desain yang lebih teknis.

e. Implementasi Sistem

Pada tahap implementasi, rancangan yang dihasilkan pada tahap perancangan sistem diwujudkan. Program komputer ditulis, dikompilasi, dan diujicoba. Selanjutnya perangkat keras dan perangkat lunak baru secara terintegrasi dipasang, di-install dan diujicoba. Semua aspek sistem informasi yang baru harus teruji dan dalam kondisi baik sebelum terjadi perpindahan (peralihan) sistem. Salah satu kegiatan pada tahapan penerapan adalah pengendalian kualitas. Semua manual dan dokumentasi sistem diperiksa ulang dan dikonsultasikan dengan para pemangku kepentingan. Para calon pengguna sistem yang baru menjalani pelatihan dan sosialisasi untuk

mengakrabkan diri. Jika calon pengguna tidak familiar dengan sistem yang baru, besar kemungkinan proses peralihan akan berjalan dengan sulit. Untuk menguji sistem informasi, data yang sesungguhnya dapat digunakan untuk uji coba. Data induk mulai dipindahkan dan ditransformasikan dari sistem lama ke sistem baru. Prosedur keamanan sistem juga menjalani uji coba untuk meyakinkan bahwa sistem baru akan terjaga integritas dan kehandalannya dari serangan maupun kesalahan yang tidak disengaja. Prosedur pemulihan kembali jika ada gangguan sistem juga diuji coba.

f. Peninjauan Ulang dan Perawatan Sistem

Tahapan peninjauan ulang dan perawatan dilakukan setelah sistem yang dibangun telah diimplementasikan dan telah berjalan. Ketika sistem sedang dioperasikan pada kondisi yang riil, sering kali ditemui situasi di mana sistem harus dimodifikasi atau diperbaiki untuk menyesuaikan dengan kondisi saat ini. Suatu sistem harus ditinjau untuk suatu perubahan sebagai akibat dari adanya permasalah (error) yang ditemukan pengguna, temuan dari hasil pemeriksaan atau audit, adanya peningkatan kapasitas kerja yang menuntut kebutuhan kapasitas penyimpanan maupun pengolahan data, perubahan lingkungan bisnis (misalnya kebijakan pemerintah), pemanfaatan kemajuan teknologi, dan lain-lain.

Kelebihan dan Kelemahan SDLC Sebagai sebuah metode pengembangan yang telah diujicoba dan digunakan dalam berbagai proyek pengembangan sistem informasi dengan berbagai ukuran dan tingkat kompleksitasnya, SDLC memiliki kelebihan dan kelemahan. **Beberapa kelebihan SDLC** sebagai sebuah metode pengembangan sistem (Avison & Fitzgerald, 2006) adalah:

a. adanya dokumentasi standar yang dapat mempermudah para pemangku kepentingan dalam berkomunikasi. dokumentasi juga menjamin bahwa spesifikasi sistem yang dikembangkan sesuai dengan kebutuhan pengguna;

b. dokumentasi yang lengkap juga memudahkan edukasi bagi pengguna sistem;

c. SDLC memiliki fase-fase dengan batasan dan hasil kegiatan yang jelas. Fase-fase tersebut mempermudah dalam mengelola proyek pengembangan sistem karena tiap fase didefinisikan dengan jelas. Kemajuan dan pencapaian proyek dapat dilihat dan dikendalikan dengan mudah karena sudah terbagi dalam tahapan yang jelas.

Kelemahan SDLC tersebut antara lain berikut ini (Avison & Fitzgerald, 2006).

a. Hanya mampu memenuhi kebutuhan manajemen tingkat bawah.
b. Pemodelan proses yang tidak dinamis.
c. Perancangan yang tidak fleksibel.
d. Munculnya ketidakpuasan pengguna.
e. Permasalahan dengan pendekatan ideal.

B. RAGAM METODOLOGI PENGEMBANGAN SISTEM

Menurut Avison dan Fitzgerald (2006), definisi dari metodologi pengembangan sistem informasi atau yang sering disebut sebagai information systems development method

(ISDM) adalah kumpulan prosedur, teknik, alat, dan alat bantu pendokumentasian yang membantu para pengembang membangun sistem informasi.

Beynon-Davis (2003) mempunyai pandangan lain tentang pengelompokan metodologi pengembangan sistem informasi. Dalam pandangan Beynon-Davis, metodologi pengembangan sistem informasi dapat dibagi menjadi tiga kelompok utama.

1. **Metode terstruktur (structured methods)** Menggunakan model linear dalam proses pengembangan. Setiap tahapan diidentifikasi dengan jelas, termasuk input dan output dari setiap tahapan. Pemodelan data dan proses dilakukan dengan kerangka kerja yang terstruktur.

   Contoh:

   a. STRADIS (Structured Analysis, Design and Implementation of Information System)

   b. SSAD (Structured System Analysis and Design)

2. **Metode Rapid Application Development (RAD)**

   Menggunakan model iterasi di dalam proses pengembangan dan secara umum menspesifikasikan tahapan level tinggi (high-level phase) berdasar beberapa bentuk prototype. Metode RAD secara umum dapat disesuaikan dengan situasi yang ada karena tidak memberikan detail teknik yang digunakan. Contoh: DSDM (Dynamic Systems Development Method)

3. **Metode berorientasi obyek (object-oriented methods)**

   Fokus pada penggunaan obyek secara konsisten mulai dari tahap analisis, perancangan, sampai implementasi sistem informasi. Proses pengembangan dengan pendekatan berorientasi obyek biasanya memanfaatkan model kontingensi, di mana kadang-kadang menggunakan model linear, kadangkandang menggunakan model iteratif. Contoh: a. UML (Unified Modelling Language) b. RUP (Rational Unified Process)

Metodologi pengembangan sistem informasi juga dapat diklasifikasikan menjadi:

1. metodologi pengembangan sistem melalui pendekatan siklus hidup pengembangan sistem (system development life cycle/SDLC) (Sulianta, 2017).

Contoh: Waterfall, RAD (Rapid Application Development), Prototyping, Spiral, Incremental, Agile Development;

2. metodologi pengembangan sistem melalui metode alternatif (Jogiyanto, 2010) Contoh: Paket, Outsourcing, End-User Development.

C. PENTINGNYA METODOLOGI PENGEMBANGAN SISTEM

Kesalahan dalam estimasi dapat mengakibatkan penyelesaian suatu proyek sistem menjadi mundur dari jadwal yang telah ditetapkan dan disepakati. Terjadi antrian pada proyek pengembangan aplikasi lainnya yang akan diselesaikan.

Adanya metodologi pengembangan sistem secara formal akan mendorong terjadinya hal-hal berikut (Avison & Fitzgerald, 2006).

1. Meningkatnya kesadaran akan perlunya analisa permasalahan dan kebutuhan sistem, serta perancangan sistem yang lebih baik sebelum sistem dibuat.

2. Terciptanya solusi sistem yang terintegrasi bagi manajemen organisasi yang besar dan kompleks.

3. Meningkatnya kesadaran akan perlunya suatu metode yang dapat memandu para pengembang sistem untuk menghasilkan aplikasi dan sistem informasi yang berkualitas.

Alasan rasional lainnya mengenai perlunya menggunakan metodologi dalam pengembangan sistem dikemukakan oleh Avison dan Fitzgerald (2006).

1. Menghasilkan Produk Akhir Berupa Sistem Yang Lebih Baik, dengan beberapa indikator:

a. waktu pengembangan yang cepat dan ekonomi,

b. sistem yang dikembangkan dapat memenuhi kebutuhan fungsional bagi pengguna,

c. mudah dipelajari dan dioperasikan,

d. tersedia dokumentasi yang memadai untuk membantu komunikasi antar pengguna, pengembang sistem, dan pihak manajemen,

e. modular, sehingga walaupun terintegrasi satu sama lainnya jika terjadi kerusakan atau modifikasi pada suatu komponen, pengaruhnya kecil terhadap komponen lain atau sistem itu sendiri,

f. sistem yang dikembangkan kompatibel dengan sistem lainnya di dalam maupun di luar organisasi,

g. dapat dijalankan di berbagai platform teknologi yang berbeda, h. sistem yang fleksibel (mudah dimodifikasi),

i. mudah dipelihara,

j. sistem mampu bertahan dalam kondisi tertentu yang tidak menguntungkan (fail-safe dan fault-tolerant),

k. aman.

2. Kematangan Proses Pengembangan Sistem

3. Standardisasi Proses Pengembangan Sistem

Studi Kasus (Diskusi)

PT. Bangkit Jaya adalah salah satu dari perusahaan produsen makanan ringan yang berdomisili di Kalimantan. Kegiatan operasional perusahaan mulai mengalami kendala akibat melimpahnya data transaksi harian. Seringkali Stakeholder kesulitan untuk mengambil tindakan dikarenakan kurang sinkronnya data dari masing-masing unit yang ada di dalam perusahaan. Perusahaan merencanakan sebuah ide dengan memperbaiki kualitas penyimpanan data dan kegiatan operasional perusahaannya supaya dapat terkomputerisasi sehingga pada saat perusahaan akan mulai menerapkan metode penjualan secara online,

semua sumber daya yang ada di dalam perusahaan sudah siap. Input data selama ini masih dilakukan secara manual, sehingga bisa dibayangkan sulitnya jika data-data tersebut terdiri dari ribuan data dan memerlukan proses yang lama. Masalah tersebut mendorong PT. Bangkit Jaya untuk membangun sebuah Sistem Informasi yang memungkinkan pencatatan data dapat dilakukan dengan cepat oleh semua bagian perusahaan, serta dapat memudahkan pengambilan keputusan secara lebih cepat dan tepat.

1.3 TUGAS PRAKTIKUM

Buat laporan untuk studi kasus yang terpilih pada Tugas 1 dengan susunan laporan sebagai berikut:

a. Judul perangkat lunak + metodologi pengembangan sistem yang dipilih.
b. Kondisi awal kasus lebih spesifik dari tugas 1
c. Sebutkan Kebutuhan Fungsional dan Non Fungsional PL yang akan dirancang
d. Tahapan metodologi pengembangan
e. Flowchart

MODUL 3
Flowchart (Diagram Alir)

1.1 Bahasan dan Tujuan

1.1.1 Bahasan

Membahas tentang pengertian, ragam dan cara pembuatan diagram alir sistem perangkat lunak

1.1.2 Tujuan

1. Mahasiswa memahami pengertian, ragam flowchart dan standar pembuatan diagram alir sistem perangkat lunak
2. Mahasiswa mampu membuat diagram alir sistem perangkat lunak

1.2 DASAR TEORI

Dalam pengembangan perangkat lunak, setelah proses analisis kebutuhan akan dilakukan proses desain dari perangkat lunak yang akan dibangun. Flowchart merupakan salah satu diagram alir yang akan membantu menggambarkan proses pengembangan perangkat lunak dari awal proses hingga akhir. Selain ***Flowchart***, terdapat juga ***Data Flow Diagram*** **(DFD),** dan ***Unified Modelling Language*** **(UML)** yang merupakan metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan Perangkat Lunak.

A. Pengertian flowchart

Flowchart atau diagram alir adalah diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah digambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah.

B. Fungsi flowchart

Fungsi utama dari flowchart adalah memberi gambaran jalannya sebuah program dari satu proses ke proses lainnya. Sehingga, alir program menjadi mudah dipahami oleh semua orang. Selain itu, fungsi lain dari flowchart adalah untuk menyederhanakan rangkaian prosedur agar memudahkan pemahaman terhadap informasi tersebut.

C. Jenis flowchart

Flowchart sendiri terdiri dari lima jenis, masing-masing jenis memiliki karakteristik dalam penggunaanya. Berikut adalah jenis-jenisnya:

a. Flowchart dokumen

Pertama ada flowchart dokumen (document flowchart) atau bisa juga disebut dengan paperwork flowchart. Flowchart dokumen berfungsi untuk menelusuri alur form dari satu bagian ke bagian yang lain, termasuk bagaimana laporan diproses, dicatat, dan disimpan.

b. Flowchart program

Selanjutnya kita akan membahas flowchart program. Flowchart ini menggambarkan secara rinci prosedur dari proses program. Flowchart program terdiri dari dua macam, antara lain: flowchart logika program (program logic flowchart) dan flowchart program komputer terinci (detailed computer program flowchart).

c. Flowchart proses

Flowchart proses adalah cara penggambaran rekayasa industrial dengan cara merinci dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

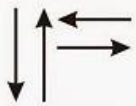
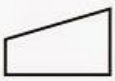
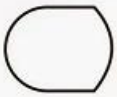
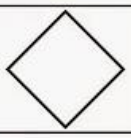
d. Flowchart sistem

Yang keempat ada flowchart sistem. Flowchart sistem adalah flowchart yang menampilkan tahapan atau proses kerja yang sedang berlangsung di dalam sistem secara menyeluruh. Selain itu flowchart sistem juga menguraikan urutan dari setiap prosedur yang ada di dalam sistem.

e. Flowchart skematik

Terakhir ada flowchart skematik. Flowchart ini menampilkan alur prosedur suatu sistem, hampir sama dengan flowchart sistem. Namun, ada perbedaan dalam penggunaan simbol-simbol dalam menggambarkan alur. Selain simbol-simbol, flowchart skematik juga menggunakan gambar-gambar komputer serta peralatan lainnya untuk mempermudah dalam pembacaan flowchart untuk orang awam.

Selanjutnya dalam implementasinya akan digunakan jenis flowchart tertentu sesuai dengan kebutuhan. Berikut ini simbol standart yang akan digunakan dalam membuat flowchart:

	**Flow Direction symbol** Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.		**Simbol Manual Input** Simbol untuk pemasukan data secara manual on-line keyboard
	**Terminator Symbol** Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan		**Simbol Preparation** Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	**Connector Symbol** Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.		**Simbol Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	**Connector Symbol** Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.		**Simbol Display** Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	**Processing Symbol** Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer		**Simbol disk and On-line Storage** Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	**Simbol Manual Operation** Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer		**Simbol magnetik tape Unit** Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	**Simbol Decision** Simbol pemilihan proses berdasarkan kondisi yang ada.		**Simbol Punch Card** Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	**Simbol Input-Output** Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya		**Simbol Dokumen** Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

1.3 PRAKTIKUM

Petunjuk pembuatan Flowchart.

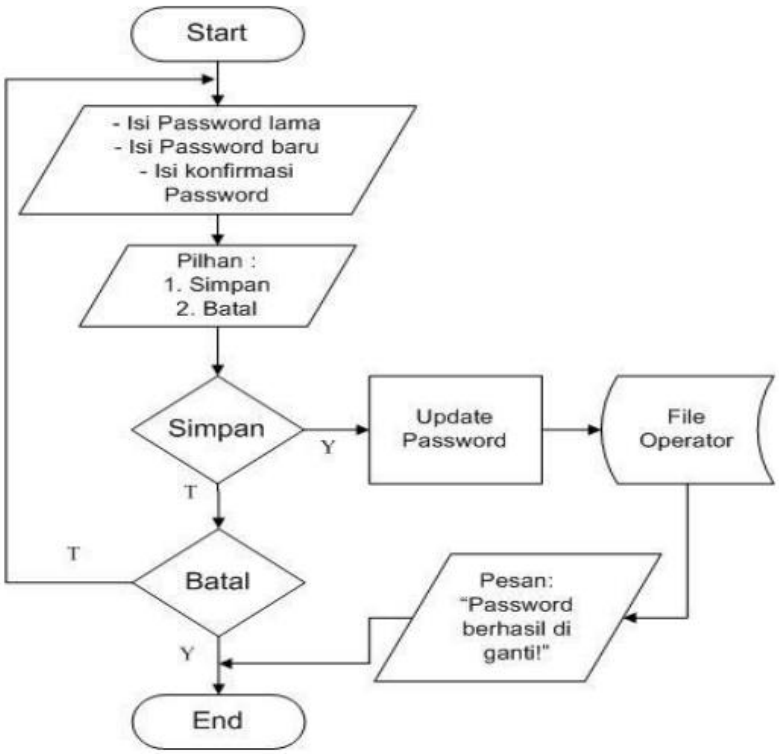
Bila seorang analis dan programmer akan membuat flowchart, ada beberapa petunjuk yang harus diperhatikan, seperti :

a) Flowchart digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.
b) Aktivitas yang digambarkan harus dapat dimengerti oleh pembacanya.
c) Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
d) Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja,
e) Setiap langkah dari aktivitas harus berada pada urutan yang benar.
f) Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
g) Gunakan simbol-simbol flowchart yang standar.

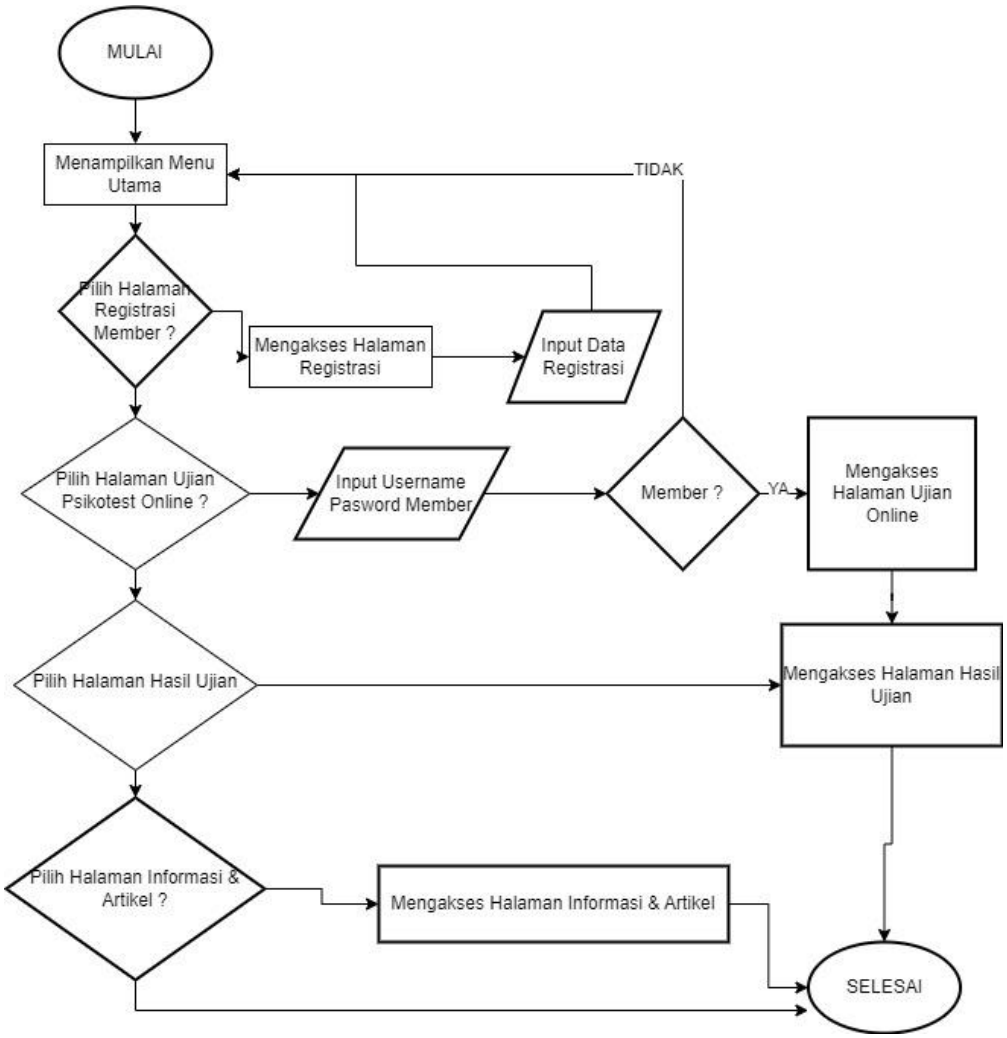
Berikut beberapa software atau tools yang dapat digunakan untuk membuat Flowchart:

a) Draw.io
b) Microsoft Visio
c) GitMind
d) Gliffy Diagram

Contoh 1. Membuat Flowchart Sistem sederhana proses penggantian password.



Contoh 2. Membuat Flowchart Sistem Sederhana Aplikasi Mobile Psikotes Online dengan 4 Halaman.



1.4 TUGAS

1. Buatlah Flowchart Sistem untuk Perangkat Lunak Anda (Tugas sebelumnya)!
2. Jelaskan secara rinci tahapan yang ada pada flowchart sistem Perangkat Lunak Anda!

MODUL 4
Entity Relationship Diagram (ERD)

4.1 Bahasan dan Tujuan

4.1.1 Bahasan

Membahas terkait basis data dan tahap-tahap yang dilakukan untuk membuat desain basis data yang digunakan pada perangkat lunak.

4.1.2 Tujuan

1. Mahasiswa memahami pengertian basis data.
2. Mahasiswa mampu memahami *Database Life Cycle* (DBLC).
3. Mamhasiswa mampu mendesain *Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM).

4.2 DASAR TEORI

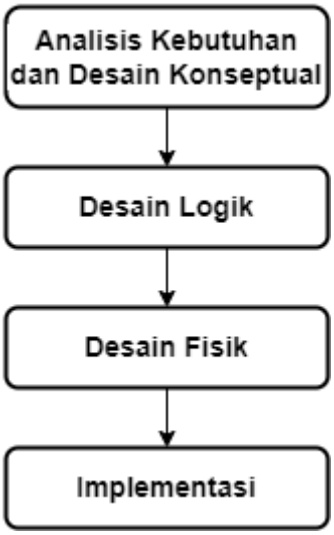
4.2.1 Basis Data

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis berarti markas/gudang atau tempat berkumpul. Sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya (Fathansyah, 2018). Basis data dapat diartikan sebagai kumpulan data yang saling berhubungan yang disimpan secara bersama tanpa pengulangan (redundansi) yang tidak perlu supaya dapat dimanfaatkan kembali dengan cepat dan mudah.

Basis data tidak diolah secara manual, melainkan ditangan dengan menggunakan sebuah perangkat lunak yang disebut *Database Management System* (DBMS). Fungsi dari DBMS adalah untuk menyimpan, mengorganisasi, dan menggunakan kembali data di dalam basis data. Pada DBMS juga terdapat mekanisme untuk pengamanan data dan pemakaian bersama (*sharing*) dari data. Pada DBMS, data yang telah disimpan dapat diakses dengan perintah-perintah tertentu. Perintahperintah yang digunakan untuk mengelola basis data mempunyai standar yang disebut dengan SQL (*Structured Query Language*).

4.2.2 *Database Life Cycle* (DBLC)

Database Life Cycle (DBLC) atau disebut juga dengan alur hidup basis data adalah alur dari perancangan sebuah basis data. Alur hidup basis data ditunjukkan pada gambar di bawah ini.



Berdasarkan gambar di atas, fase-fase DBLC terdiri dari 4 tahap yaitu:

1. **Analisis Kebutuhan atau *Requirement Analysis***

   Pada tahap ini hal-hal yang harus dilakukan yaitu:

   a. Mendefinisikan data yang terkait dengan perangkat lunak yang akan dikembangkan dan perlu disimpan. Hal tersebut biasanya dilakukan dengan melakukan wawancara kepada produsen dan pengguna data.
   b. Membuat kontrak spesifikasi basis data.
   c. Merancang *Entity Relationship Diagram* (ERD) sebagai bagian dari desain konseptual.

2. **Desain Logik (*Logical Database Design)***

   Pada tahap desain logik hal yang dilakukan adalah merancang desain *Conceptual Data Model* (CDM).

3. **Desain Fisik (*Physical Database Design)***

   Pada tahap ini perlu dibuat rancangan desain *Physical Data Model* (PDM) sebagai sebuah rancangan basis data yang siap diimplementasikan di DBMS.

4. **Implementasi**

   Tahap terakhir yaitu implementasi, pada tahap implementasi rancangan basis data yang sudah dibuat sebelumnya mulai diimplementasikan menggunakan DBMS dan *Query SQL* dirancang untuk kebutuhan akses basis data di DBMS tersebut.

4.2.3 *Entity Relationship Diagram* (ERD)

Entity Relational Diagram (ERD) adalah diagram yang digunakan untuk menggambarkan ER-Model. ER-Model yaitu suatu model yang digunakan untuk menjelaskan hubungan antar data dalam basis data. Komponen ERD yaitu **entitas**, **atribut** dan **relasi**.

a. Entitas

Entitas adalah segala sesuatu (objek) yang informasinya perlu disimpan di dalam basis data. Entitas biasanya digambarkan dengan bentuk persegi panjang.

Entitas memiliki beberapa jenis yaitu entitas kuat, entitas lemah, entitas super type dan entitas sub type.

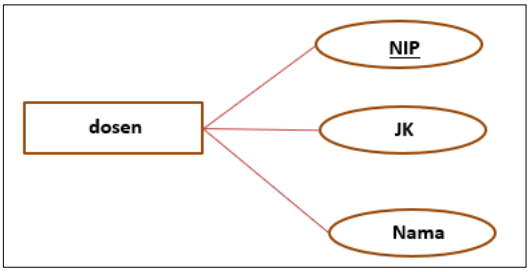
b. Atribut

Atribut merupakan ciri atau karakteristik atau keterangan yang terkait atau dimiliki oleh entitas. Atribut digambarkan dengan bentuk oval.

Jenis-jenis atribut:

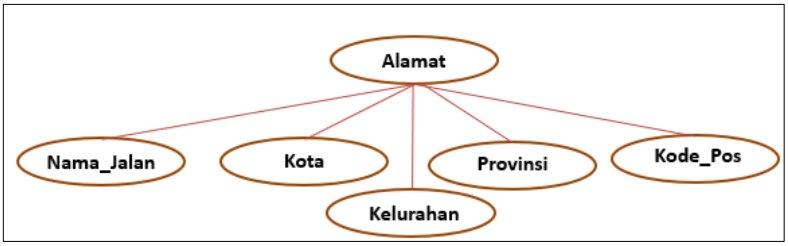
- **Atribut sederhana**

Atribut yang terdiri dari satu komponen tunggal (bersifat *independent*). Contoh: atribut NIP, JK dan Nama pada entitas dosen.



- **Atribut komposit**

Atribut yang terdiri dari beberapa atribut yang lebih mendasar (dapat diuraikan).

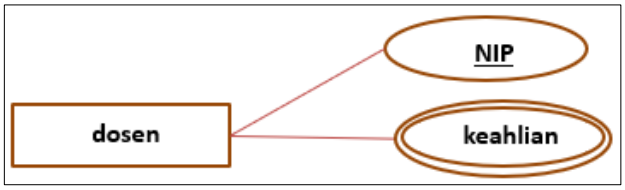


- **Atribut bernilai tunggal**

Atribut yang hanya mempunyai satu nilai untuk suatu entitas tertentu. Contoh: NIM, Nama, JK pada entitas dosen.

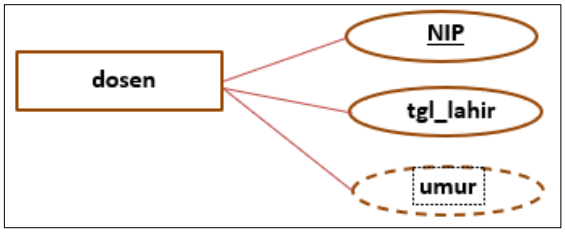
- **Atribut bernilai banyak**

Atribut yang dapat terdiri dari sekumpulan nilai untuk suatu entitas tertentu. Contoh: atribut hobi, keahlian.



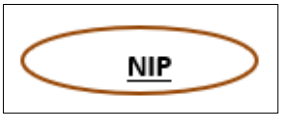
- **Atribut derivat**

Atribut yang dihasilkan dari atribut lain yang tidak berasal dari satu entitas. Contoh: atribut umur yang dihasilkan dari atribut tgl_lahir dan tanggal hari ini.



- **Atribut identitas**

Atribut yang secara unik mengidentifikasi masing-masing instan dari suatu entitas. Contoh: atribut NIP pada entitas dosen.

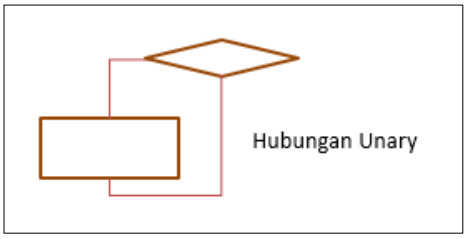


c. Relasi

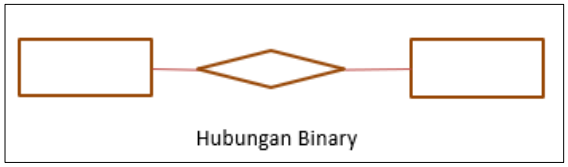
Hubungan yang terjadi antara satu atau lebih entitas. Dapat memiliki atribut, dimana terjadi adanya transaksi yang menghasilkan suatu nilai tertentu. Pada ERD relasi disimbolkan menggunakan bentuk belah ketupat dan kata yang mendeskripsikannya menggunakan kata kerja.

Jumlah entitas yang terlibat didalam suatu hubungan atau *relationship*, dibagi menjadi:

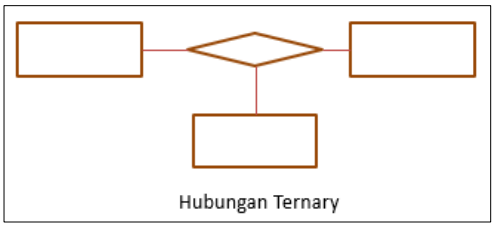
- **Hubungan Unary**



- **Hubungan Binary**



- **Hubungan Ternary**

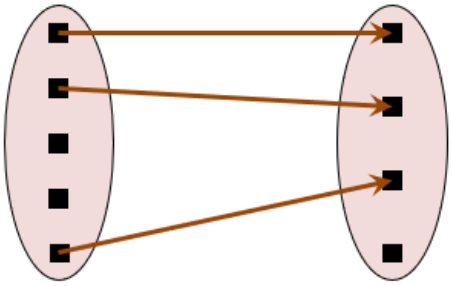


Cardinality Ratio

Cardinality ratio disebut juga dengan kardinalitas hubungan dimana kardinalitas hubungan ini menjelaskan jumlah keterhubungan satu entitas dengan entitas lainnya. Terdapat beberapa kardinalitas hubungan yaitu

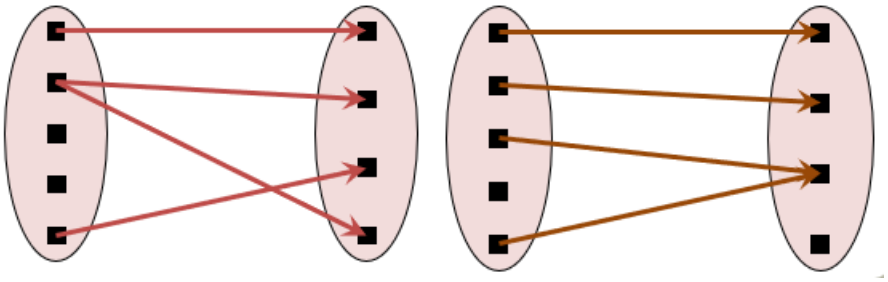
- ***One to One* (1:1)**

Kondisi dimana setiap entitas dalam hubungan memiliki satu dan hanya satu entitas pasangan.



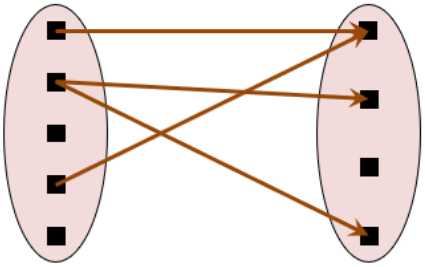
- ***One to Many* (1:N atau N:1)**

Kondisi dimana satu entitas di satu pihak dalam suatu hubungan dapat memiliki beberapa pasangan di pihak lawannya, tetapi entitas di pihak lawannya hanya boleh memiliki maksimum satu pasangan.



- ***Many to Many* (M:N)**

Kondisi hubungan dimana entitas di masing-masing sisi dari hubungan dapat memiliki beberapa pasangan di sisi yang lain.



Constraint Cardinality

Constraint cardinality atau konstrain kardinalitas merupakan batasan jumlah instan/data dari satu entitas yang dapat atau harus berasosiasi dengan setiap instan/data dari entitas yang lain. Konstrain kardinalitas dibagi menjadi 2 yaitu:

- **Kardinalitas Minimum**

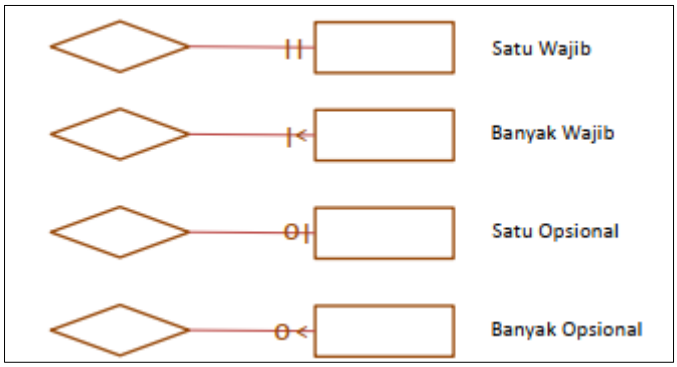
Kondisi dimana sebuah baris data di dalam suatu tabel harus dihubungkan paling tidak dengan satu baris didalam tabel yang berelasi dengannya.

- Jika Nol, berarti Opsional (Tidak Wajib).
- Jika Satu atau lebih, berarti Wajib.

- **Kardinalitas Maksimum**

Kondisi dimana sebuah baris data di dalam suatu tabel dapat dihubungkan kelebih dari satu baris pada tabel lainnya.

Konstrain kardinalitas ditunjukkan oleh gambar di bawah ini:

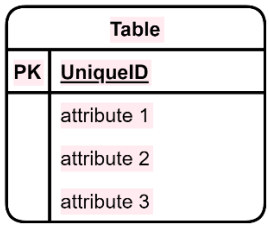
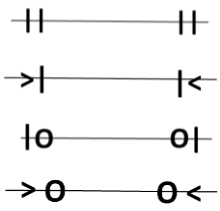


Apabila disimpulkan tahapan dalam merancang ERD yaitu:

1. Mengidentifikasikan dan menetapkan seluruh himpunan entitas yang akan terlibat.
2. Menentukan atribut-atribut dari setiap entitas.
3. Menentukan atribut *primary key* dari setiap entitas.
4. Menentukan *relationship* antar entitas.
5. Menentukan atribut-atribut dari setiap *relationship* (jika ada).
6. Menentukan *cardinality ratio*.
7. Menentukan *constraint cardinality*.

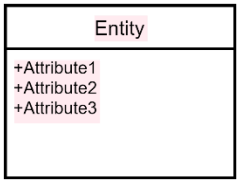
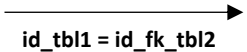
4.2.4 *Conceptual Data Model* (CDM)

Conceptual Data Model (CDM) merupakan konsep yang berkaitan dengan pandangan pengguna terhadap data yang disimpan dalam basis data. CDM dibuat dalam bentuk tabel-tabel tanpa tipe data yang menggambarkan relasi antar tabel untuk keperluan implementasi ke basis data. Berikut adalah simbol-simbol yang ada pada CDM:

Simbol	Deskripsi
 Entitas atau Tabel	Entitas atau tabel yang menyimpan data dalam basis data
 Relasi	Relasi antar tabel yang terdiri dari nama relasi dan konstrain kardinalitas.

4.2.5 *Physical Data Model* (PDM)

Physical Data Model (PDM) adalah model yang menggunakan sejumlah tabel untuk menggambarkan data serta hubungan antar data. PDM menerangkan detail dari bagaimana data disimpan di dalam basis data. PDM merupakan bentuk fisik perancangan basis data yang sudah siap diimplementasikan ke dalam DBMS sehingga nama tabel yang digunakan merupakan nama asli tabel yang akan diimplementasikan ke dalam DBMS. Berikut simbol-simbol yang ada pada PDM:

Simbol	Deskripsi
 Entitas atau Tabel	Entitas atau tabel yang menyimpan data dalam basis data
 Relasi	Relasi antar tabel yang terdiri dari persamaan antara *primary key* tabel yang diacu dengan kunci yang menjadi referensi acuan di tabel yang lain.

4.3 PRAKTIKUM

Sistem inventori barang merupakan sebuah sistem untuk mengelola proses inventori barang secara sederhana yang diperlukan untuk mendata barang. Sistem ini meliputi memasukkan data serta mengelola data pemasok, barang, petugas, dan pihak manajemen beserta proses-proses pelaporannya. Aturan inventori barang yang harus diatasi pada sistem inventori barang yang akan dimodelkan adalah bahwa sebuah barang memiliki satu pemasok.

Pengguna dari Sistem Inventori Barang adalah administrator/petugas dan pihak manajemen. Fungsi-fungsi dari Sistem Inventori Barang adalah sebagai berikut:

1. Memproses login untuk pengguna sistem.
2. Mengelola data pengguna (*user*) oleh Administrator/Petugas yang meliputi:
   a. Memasukkan data pengguna.
   b. Mengubah data pengguna.
   c. Menghapus data pengguna.
   d. Mencari atau melihat data pengguna.
3. Mengelola data pemasok (*supplier*)
   a. Memasukkan data pemasok.
   b. Mengubah data pemasok.
   c. Menghapus data pemasok.
   d. Mencari atau melihat data pemasok.
4. Mengelola data barang
   a. Memasukkan data barang.
   b. Mengubah data barang.
   c. Menghapus data barang.
   d. Mencari atau melihat data barang.
5. Memproses laporan untuk pihak manajemen

4.3.1. Definisi Entitas dan Atribut

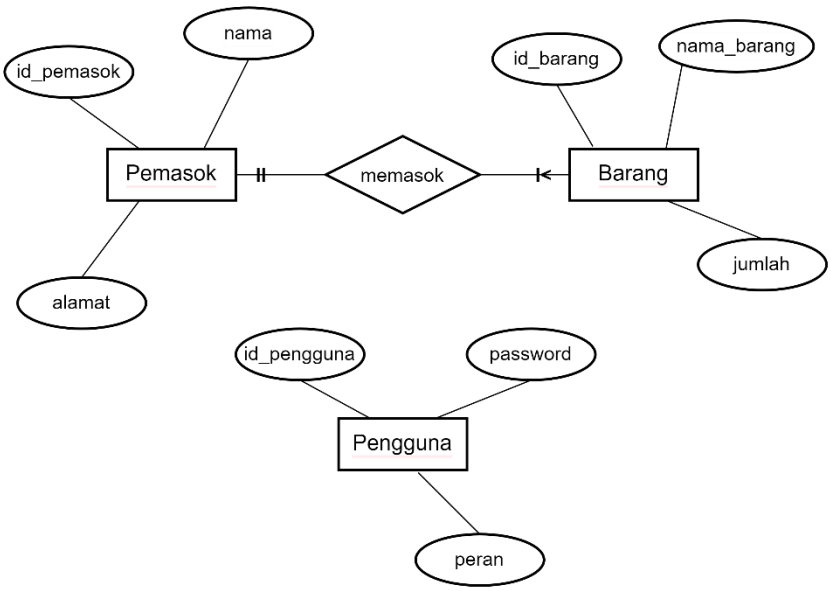
No	Entitas	Atribut
1	**Pengguna** Entitas yang menyimpan data pengguna	**id_pengguna** atribut yang menjadi identitas dari pengguna
		password atribut password yang digunakan untuk proses login
		peran

		atribut jenis pengguna (administrator atau manajemen)
2	**Pemasok** Entitas yang menyimpan data pemasok	**id_pemasok** atribut yang menjadi identitas dari pemasok
		nama_barang atribut nama pemasok
		alamat atribut alamat pemasok
3	**Barang** Entitas yang menyimpan data barang	**id_barang** atribut identitas barang
		nama_barang atribut nama barang
		jumlah atribut jumlah barang

4.3.2 Definisi Relasi

No	Relasi	Deskripsi
1	Memasok	Relasi antara entitas Pemasok dengan entitas Barang. Kardinalitas antara entitas Pemasok dengan entitas Barang adalah *one to many* karena seorang pemasok dapat memasok banyak barang sedangkan sebuah barang hanya memiliki satu pemasok.

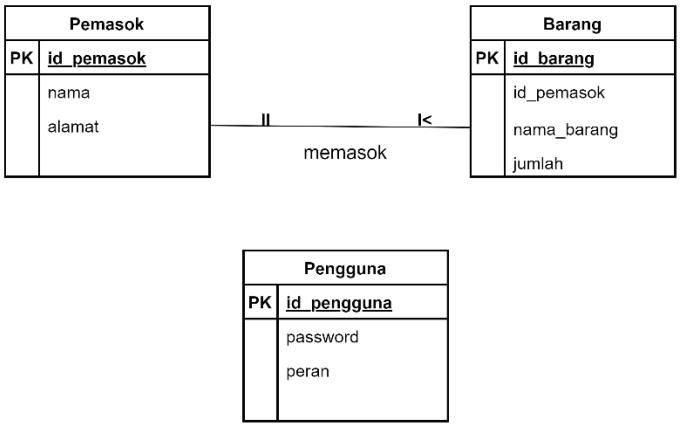
4.3.3 Diagram ER



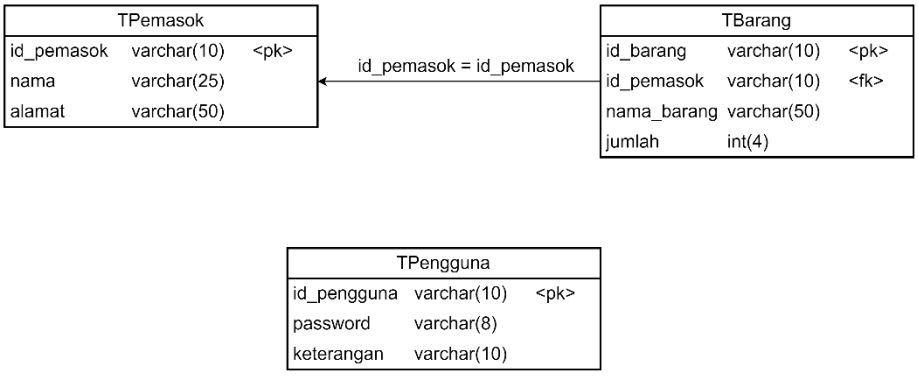
Berdasarkan gambar diagram ER di atas:

a. Lakukan perbaikan dari kekurangan yang ada pada ER diagram di atas.
b. Jelaskan konstrain kardinalitas yang terjadi di dalam ER diagram di atas.

4.3.4 CDM



4.3.5 PDM



4.4 TUGAS PROGRESS PROJECT AKHIR SEMESTER

Diskusikan bersama kelompok dan buatlah rancangan ER Diagram, CDM dan PDM untuk Perangkat Lunak Anda (melanjutkan tugas sebelumnya).

MODUL 4
Data Flow Diagram (DFD)

4.1 Bahasan dan Tujuan

4.1.1 Bahasan

Memahami Konsep Pendekatan Perancangan Terstruktur dengan Salah Satu Alat Bantunya DFD, serta Dapat Menggunakan DFD Secara Tepat.

4.1.2 Tujuan

1. Mahasiswa mengetahui konsep DFD serta bagaimana cara menggunakannya
2. Mahasiswa dapat menentukan kapan menggunakan secara tepat berdasarkan kelebihan dan kekurangan DFD
3. Mahasiswa diharapkan dapat membuat model sistem yang akan mereka kembangkan dengan DFD

4.2 DASAR TEORI

4.2.1 Konsep Perancangan Terstruktur

Karena banyak terjadi permasalahan-permasalahan di pendekatan klasik, maka kebutuhan akan pendekatan pengembangan sistem yang lebih baik mulai terasa dibutuhkan. Sayangnya sampai sekarang masih banyak orang yang tidak menyadari bahwa hanya dengan mengikuti tahapan di life cycle saja tidak akan membuat pengembangan sistem informasi menjadi berhasil. Oleh karena itu diperlukan suatu pendekatan pengembangan sistem yang baru yang dilengkapi dengan beberapa alat dan teknik supaya membuatnya berhasil. Pendekatan ini yang dimulai dari awal tahun 1970 disebut dengan pendekatan terstruktur (structured approach). Pendekatan terstruktur dilengkapi dengan alat-alat (tools) dan teknik-teknik (techniques) yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir dari sistem yang dikembangkan akan didapatkan sistem yang strukturnya didefinisikan dengan baik dan jelas.

Konsep pengembangan sistem terstruktur bukan merupakan konsep yang baru. Teknik perakitan di pabrik-pabrik dan perancangan sirkuit untuk alat-alat elektronik adalah dua contoh dari konsep ini yang banyak digunakan di industri-industri. Konsep ini memang relatif masih baru digunakan dalam mengembangkan sistem informasi untuk dihasilkan produk sistem yang memuaskan pemakainya. Melalui pendekatan terstruktur, permasalahan-permasalahan yang komplek di organisasi dapat dipecahkan dan hasil dari sistem akan mudah untuk dipelihara, fleksibel, lebih memuaskan pemakainya, mempunyai dokumentasi yang baik, tepat pada waktunya, sesuai dengan anggaran biaya pengembangannya, dapat meningkatkan produktivitas dan kualitasnya

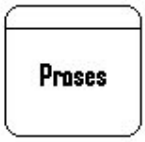
akan lebih baik (bebas kesalahan). Salah satu tools dan teknik dalam pengembangan sistem terstruktur adalah menggunakan DFD (Data Flow Diagram = Diagram Arus Data, DAD).

4.2.2 Data Flow Diagram (DFD)

Ide dari suatu bagan untuk mewakili arus data dalam suatu sistem bukanlah hal yang baru. Pada tahun 1967, Martin dan Estrin memperkenalkan suatu algorima program dengan menggunakan simbol lingkaran dan panah untuk mewakili arus data. E. Yourdan dan L. L. Constantine juga menggunakan notasi simbol ini untuk menggambarkan arus data dalam perancangan program. G.E. Whitehouse tahun 1973 juga menggunakan notasi semacam ini untuk membuat model-model sistem matematika. Penggunaan notasi dalam diagram arus data ini sangat membantu sekali untuk memahami suatu sistem pada semua tingkat kompleksitasnya seperti yang diungkapkan oleh Chris Gane dan Trish Sarson. Pada tahap analisis, penggunaan notasi ini sangat membantu sekali di dalam komunikasi dengan pemakai sistem untuk memahami sistem secara logika. Diagram yang menggunakan notasi-notasi ini untuk menggambarkan arus dari data sistem sekarang dikenal dengan nama diagram arus data (data flow diagram, DFD).

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir (misalnya lewat telpon, surat dan sebagainya) atau lingkungan fisik dimana data tersebut akan disimpan (misalnya file kartu, microfile, harddisk, tape, diskette dan lain sebagainya). DFD merupakan alat yang cukup populer sekarang ini, karena dapat menggambarkan arus data di dalam sistem dengan terstruktur dan jelas. Lebih lanjut DFD juga merupakan dokumentasi dari sistem yang baik.

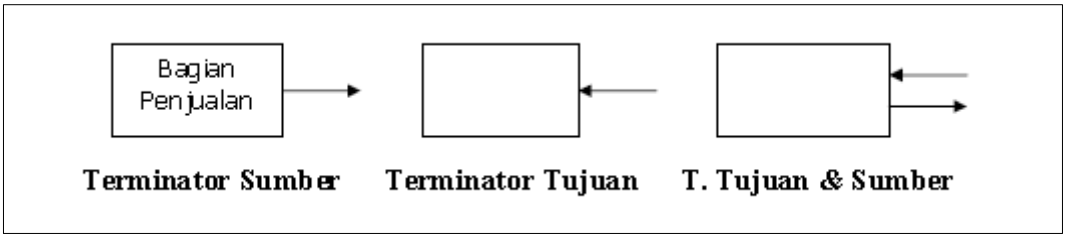
A. Komponen DFD

Gane/Sarson	Yourdon/De Marco	Keterangan
		Entitas eksternal, dapat berupa orang/unit terkait yang berinteraksi dengan sistem tetapi diluar sistem
		Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diidentifikasikan.
		Aliran data dengan arah khusus dari sumber ke tujuan
		Penyimpanan data atau tempat data direfer oleh proses.

B. Komponen Entitas Ekternal/ Terminator

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya terminator dikenal dengan nama entitas luar (external entity). Terdapat dua jenis terminator :

1. Terminator Sumber (source) : merupakan terminator yang menjadi sumber.
2. Terminator Tujuan (sink) : merupakan terminator yang menjadi tujuan data



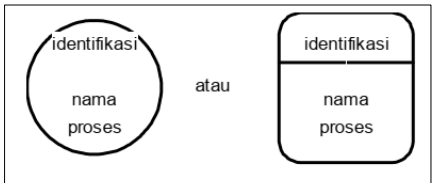
Entitas Ekternal/ Terminator dapat berupa orang, sekelompok orang, organisasi, departemen di dalam organisasi, atau perusahaan yang sama tetapi di luar kendali sistem yang sedang dibuat modelnya. Terminator dapat juga berupa departemen, divisi atau sistem diluar sistem yang berkomunikasi dengan sistem yang sedang dikembangkan.

Ada tiga hal penting yang harus diingat tentang terminator :

1. Terminator merupakan bagian/lingkungan luar sistem. Alur data yang menghubungkan terminator dengan berbagai proses sistem, menunjukkan hubungan sistem dengan dunia luar.
2. Profesional sistem tidak dapat mengubah isi atau cara kerja organisasi, atau prosedur yang berkaitan dengan terminator.
3. Hubungan yang ada antar terminator yang satu dengan yang lain tidak digambarkan pada DFD.

C. Komponen Proses

Komponen proses menggambarkan bagian dari sistem yang mentransformasikan input menjadi output. **Proses diberi nama untuk menjelaskan proses/kegiatan apa yang sedang/akan dilaksanakan**. Pemberian nama proses dilakukan dengan menggunakan kata kerja transitif (kata kerja yang membutuhkan obyek), seperti menghitung gaji, mencetak krs, menghitung jumlah sks



Gambar. Notasi proses di DFD

- Setiap proses harus diberi penjelasan yang lengkap meliputi berikut ini :

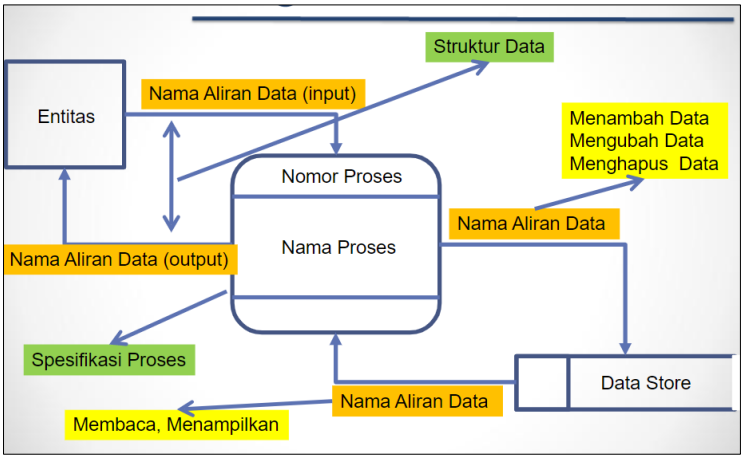
1. Identifikasi proses

Identifikasi ini umumnya berupa suatu angka yang menunjukkan nomor acuan dari

proses dan ditulis pada bagian atas di simbol proses.

2. Nama proses

Nama proses menunjukkan apa yang dikerjakan oleh proses tersebut. Nama dari proses harus jelas dan lengkap menggambarkan kegiatan prosesnya. Nama dari proses biasanya berbentuk suatu kalimat diawali dengan kata kerja (misalnya menghitung, membuat, membandingkan, memverifikasi, mempersiapkan, merekam dan lain sebagainya). Nama dari proses diletakkan di bawah identifikasi proses di simbol proses.

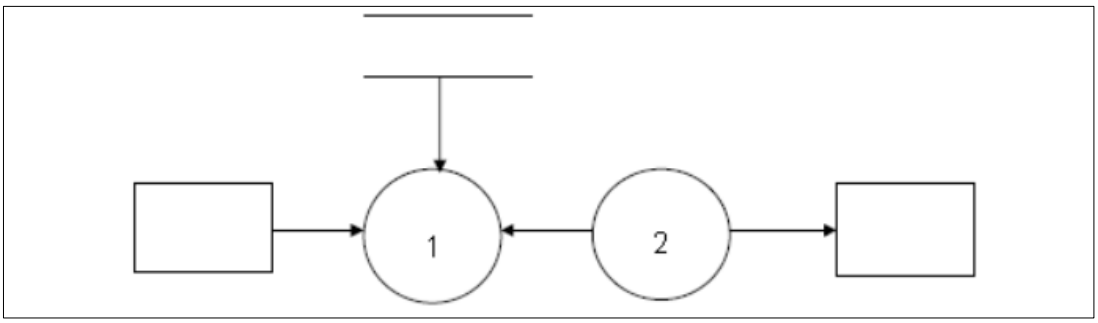


Gambar : DFD

- Ada beberapa hal yang perlu diperhatikan tentang proses :

1. Proses harus memiliki input dan output.
2. Proses dapat dihubungkan dengan komponen terminator, data store atau proses melalui alur data.
3. Sistem/bagian/divisi/departemen yang sedang dianalisis oleh profesional sistem digambarkan dengan komponen proses.

- Berikut ini merupakan suatu contoh proses yang salah :



Umumnya kesalahan proses di DFD adalah :

1. Proses mempunyai input tetapi tidak menghasilkan output. Kesalahan ini disebut dengan **black hole** (lubang hitam), karena data masuk ke dalam proses dan lenyap tidak berbekas seperti dimasukkan ke dalam lubang hitam (lihat proses 1).

2. Proses menghasilkan output tetapi tidak pernah menerima input.Kesalahan ini disebut dengan **miracle** (ajaib), karena Ajaib dihasilkan output tanpa pernah menerima input (lihat proses 2).

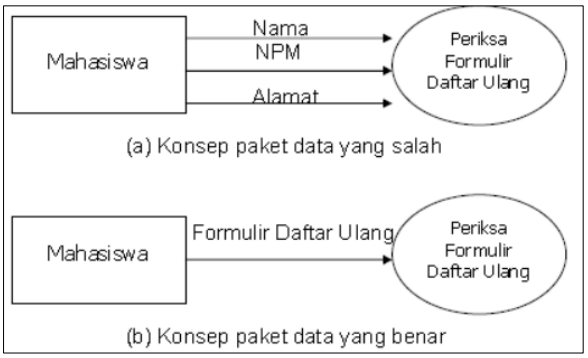
D. Aliran data/alur data

Suatu alur data / data flow dengan anak panah, yang menunjukkan arah menuju ke dan keluar dari suatu proses. Alur data ini digunakan untuk menerangkan perpindahan data atau paket data/informasi dari satu bagian sistem ke bagian lainnya. Alur data perlu diberi nama sesuai dengan data/informasi yang dimaksud, biasanya pemberian nama pada alur data dilakukan dengan menggunakan kata **benda**, contohnya **Laporan Penjualan.**

- **Konsep dalam alur data**

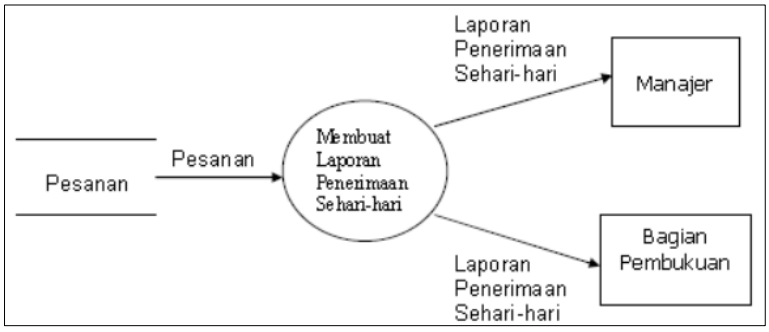
1. Konsep Paket Data (Packets of Data)

Apabila dua data atau lebih mengalir dari suatu sumber yang sama menuju ke tujuan yang sama dan mempunyai hubungan, dan harus dianggap sebagai satu alur data tunggal, karenamdata itu mengalir bersama-sama sebagai satu paket.



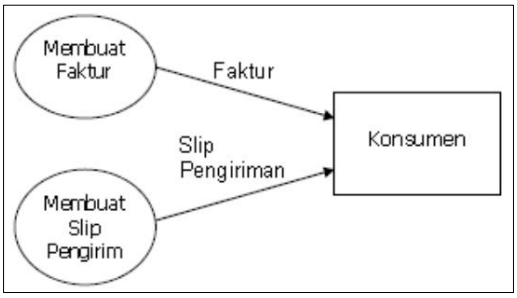
2. Konsep Alur Data Menyebar (Diverging Data Flow)

Alur data menyebar menunjukkan sejumlah tembusan paket data yang yang berasal dari sumber yang sama menuju ke tujuan yang berbeda, atau paket data yang kompleks dibagi menjadi beberapa elemen data yang dikirim ke tujuan yang berbeda, atau alur data ini membawa paket data yang memiliki nilai yang berbeda yang akan dikirim ke tujuan yang berbeda.



3. Konsep Alur Data Mengumpul (Converging Data Flow)

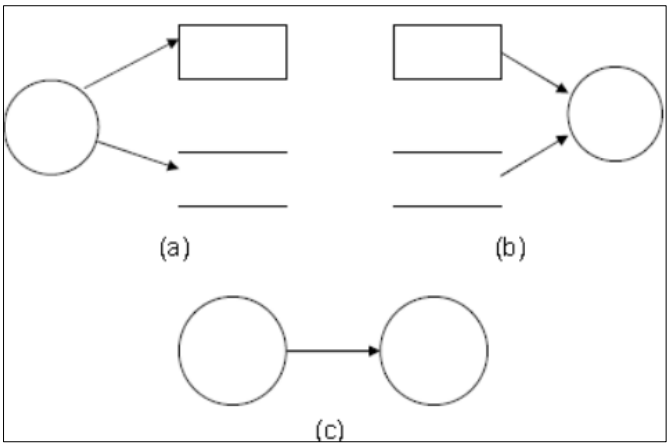
Beberapa alur data yang berbeda sumber bergabung bersama-sama menuju ke tujuan yang sama



4. Konsep Sumber atau Tujuan Alur Data

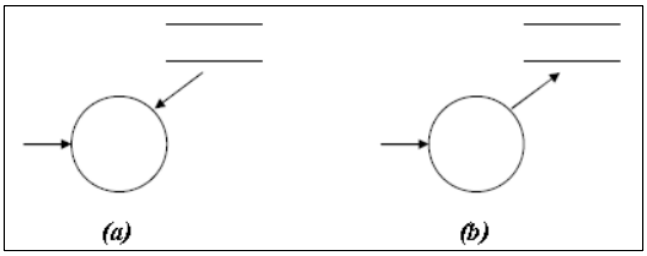
Semua alur data harus minimal mengandung satu proses. Maksud kalimat ini adalah :

1. Suatu alur data dihasilkan dari suatu proses dan menuju ke suatu data store dan/atau terminator
2. Sutu alur data dihasilkan dari suatu data store dan/atau terminator dan menuju ke suatu proses
3. Suatu alur data dihasilkan dari suatu proses dan menuju ke suatu proses



E. Data Store

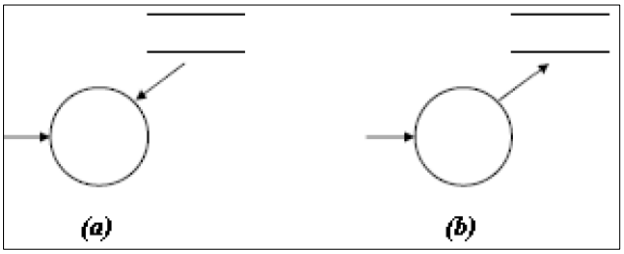
Komponen ini digunakan untuk membuat model sekumpulan paket data dan **diberi nama dengan kata benda jamak**, misalnya **Mahasiswa**



Suatu data store dihubungkan dengan alur data **hanya pada komponen proses**, tidak dengan komponen DFD lainnya. Alur data yang menghubungkan data store dengan suatu proses mempunyai pengertian sebagai berikut :

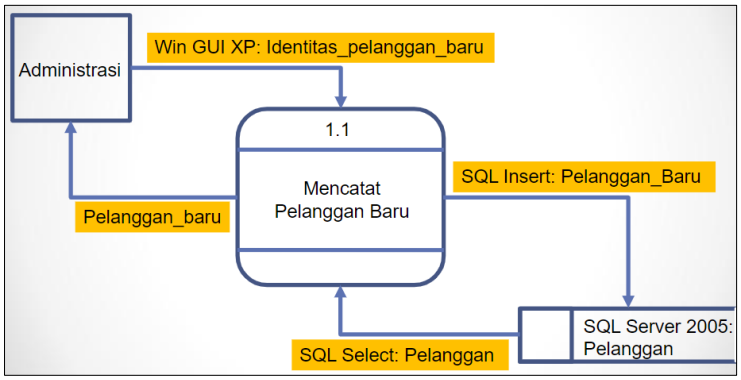
a. Alur data dari data store yang berarti sebagai pembacaan atau pengaksesan satu paket tunggal data, lebih dari satu paket data, sebagian dari satu paket tunggal data, atau sebagian dari lebih dari satu paket data untuk suatu proses

b. Alur data ke data store yang berarti sebagai pengupdatean data, seperti menambah satu paket data baru atau lebih, menghapus satu paket atau lebih, atau mengubah/memodifikasi satu paket data atau lebih



F. Bentuk DFD

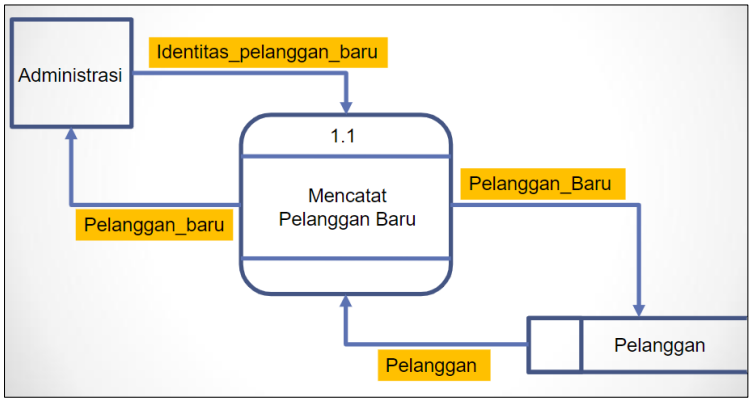
Terdapat 2 bentuk DFD, yaitu DFD fisik (**Physical Data Flow Diagram**) dan DFD logika (**Logical Data Flow Diagram).** DFD fisik lebih menekankan pada bagaimana proses dari sistem diterapkan sedang DFD logika lebih menekankan proses-proses apa yang terdapat di sistem. **Physical Data Flow Diagram** (PDFD) lebih tepat digunakan untuk menggambarkan sistem yang ada (sistem yang lama). Penekanan dari PDFD adalah bagaimana proses-proses dari sistem diterapkan (dengan cara apa, oleh siapa dan dimana), termasuk proses-proses manual. Dengan menggunakan PDFD, bagaimana proses sistem yang ada akan lebih dapat digambarkan dan dikomunikasikan kepada pemakai sistem, sehingga analis sistem akan dapat memperoleh gambaran yang jelas bagaimana sistem tersebut bekerja.



Gambar: Physical Data Flow Diagram

Logical Data Flow Diagram (LDFD) lebih tepat digunakan untuk menggambarkan sistem yang akan diusulkan (sistem yang baru). LDFD tidak menekankan pada bagaimana sistem diterapkan, tetapi penekanannya hanya pada logika dari kebutuhan-kebutuhan sistem, yaitu proses-proses apa secara logika yang dibutuhkan oleh sistem. Karena sistem

yang diusulkan belum tentu diterima oleh pemakai sistem dan biasanya sistem yang diusulkan terdiri dari beberapa alternatif, maka penggambaran sistem secara logikaterlebih dahulu tanpa berkepentingan dengan penerapannya secara fisik akan lebih mengena dan menghemat waktu penggambarannya dibandingkan dengan PDFD. Untuk sistem komputerisasi, penggambaran LDFD yang hanya menunjukkan kebutuhan proses dari sistem yang diusulkan secara logika, biasanya proses-proses yang digambarkan hanya merupakan proses-proses secara komputer saja.



Gambar: Logical Data Flow Diagram

G. Syarat Pembuatan DFD

Pedoman bagaimana menggambar DFD baik PDFD ataupun LDFD adalah sebagai berikut ini:

1. Identifikasikan terlebih dahulu semua kesatuan luar (external entity) yang terlibat di sistem. Kesatuan luar ini merupakan kesatuan (entity) di luar sistem, karena di luar bagian pengolahan data (sistem informasi). Kesatuan luar ini merupakan sumber arus data ke sistem informasi serta tujuan penerima arus data hasil dari proses sistem informasi, shingga merupakan kesatuan di luar sistem informasi.
2. Identifikasikan semua input dan output yang terlibat dengan kesatuan luar.

Kesatuan luar	Input	Output
Langganan	Order langganan	-
Bagian gudang	-	Tembusan permintaan persedian
Bagian pengiriman	Tembusan jurnal	Faktur, tembusan kredit dan tembusan jurnal
Manajer kredit	-	Status piutang

3. Gambarlah terlebih dahulu suatu **diagram konteks (context diagram).** Untuk memudahkan dalam penalaran level DFD dapat menggunakan diagram **dekomposisi (decomposition diagram).** DFD merupakan alat untuk structured analysis. Pendekatan terstruktur ini mencoba untuk menggambarkan sistem pertama kali secara garis besar (disebut dengan top level) dan memecah-mecahnya menjadi bagian yang lebih terinci (disebut dengan lower level). DFD yang pertama kali digambar adalah level teratas

(top level) dan diagram ini disebut **context diagram**. Dari context diagram ini kemudian akan Digambar dengan lebih terinci lagi yang disebut dengan overview diagram (level 0). Tiap- tiap proses di overview diagram akan Digambar secara lebih terinci lagi dan disebut dengan level 1. Tiap-tiap proses di level 1 akan digambar kembali dengan lebih terinci lagi dan disebut dengan level 2 dan seterusnya sampai tiap-tiap proses tidak dapat digambar lebih terinci lagi.

- **Diagram konteks adalah** diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke dalam sistem atau output dari sistem yang memberi gambaran tentang keseluruhan sistem.
- **Diagram dekomposisi adalah** diagram yang digunakan untuk menggambarkan dekomposisi sebuah sistem disebut juga bagan hierarki menunjukkan dekomposisi fungsional top-down dan struktur sistem. Diagram dekomposisi merupakan alat perencanaan untuk model proses yang lebih detail, yaitu diagram aliran data.

H. Perbedaan DFD Dengan Bagan Alir

DFD sangat berbeda dengan bagan alir (flow-chart). Perbedaannya adalah sebagai berikut :

1. proses di DFD dapat beroperasi secara paralel, sehingga beberapa proses dapat dilaku- kan serentak. Hal ini merupakan kelebihan DFD dibandingkan dengan bagan alir yang cenderung hanya menunjukkan proses yang urut. Kenyataannya kegiatan- kegiatan proses dapat dilakukan secara tidak urut, yaitu secara paralelnatau serentak, sehingga DFD dapat menggambarkan proses semacam ini dengan lebih mengena.
2. DFD lebih menunjukkan arus data di suatu sistem, sedang bagan alir sistem lebih menunjukkan arus dari prosedur dan bagan alir program lebih menunjukkan arus dari algoritma.
3. DFD tidak menunjukkan proses perulangan (loop) dan proses keputusan (decision), sedang bagan alir menunjukkannya.

I. Keterbatasan DFD

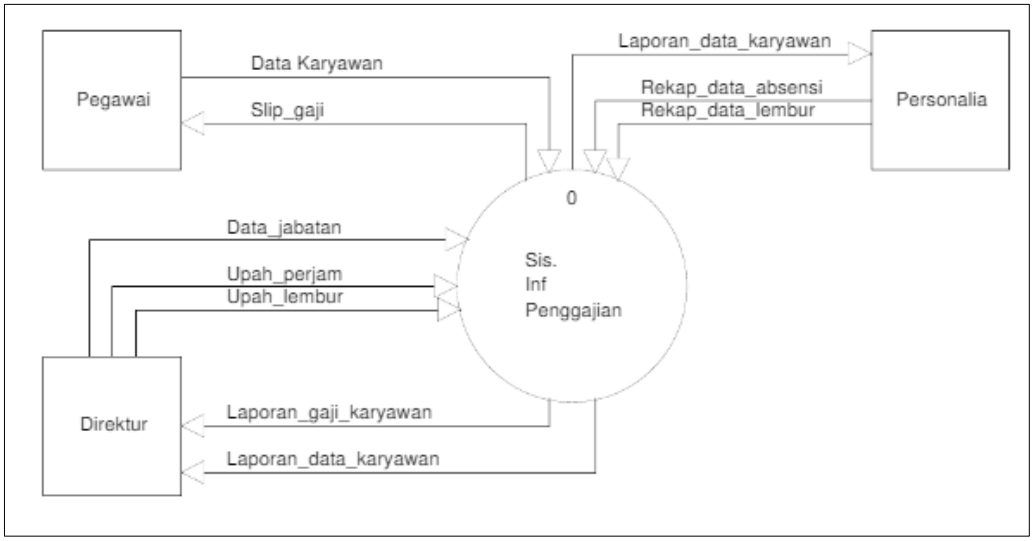
Walaupun DFD mempunyai kebaikan-kebaikan, yaitu dapat menggambarkan sistem secara terstruktur dengan memecah-mecah menjadi level lebih rendah (decomposition), dapat menunjukkan arus data di sistem, dapat menggambarkan proses paralel di sistem, dapat menunjukkan simpanan data, dapat menunjukkan kesatuan luar, tetapi DFD juga mempunyai keterbatasan. Keterbatasan DFD adalah sebagai berikut :

1. DFD tidak menunjukkan proses perulangan (loop)
2. DFD tidak menunjukkan proses keputusan (decision)
3. DFD tidak menunjukkan proses perhitungan

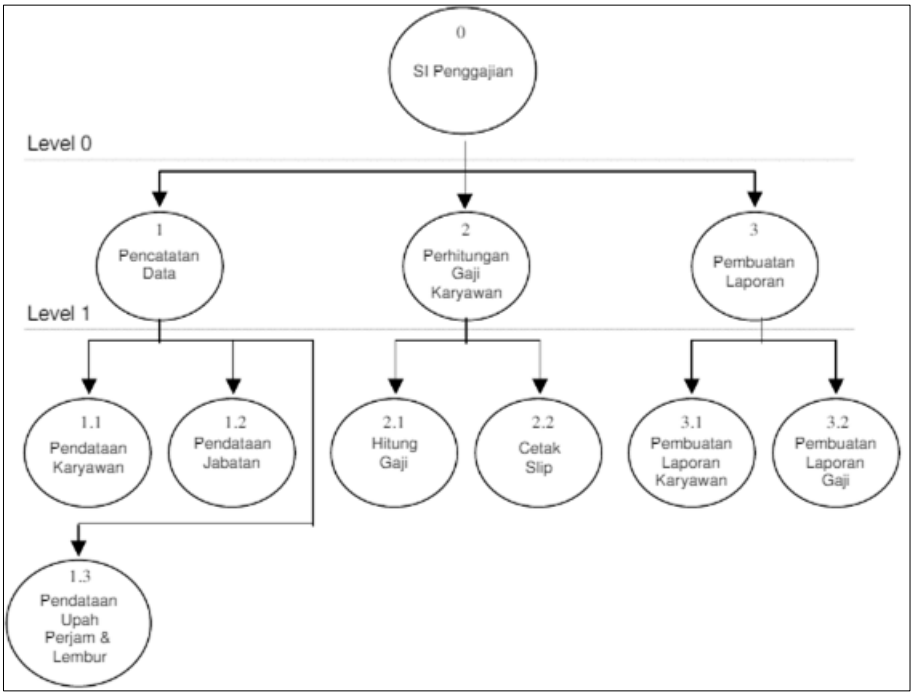
4.3 PRAKTIKUM

Terdapat Sistem Informasi Penggajian dengan spesifikasi sebagai berikut:

1. Identifikasi Sistem
   1.1. Identifikasi Data dan Informasi
      1.1.1. Identifikasi Data
         a. Data Karyawan
         b. Rekap Data Absensi
         c. Rekap Data Lembur
         d. Data Jabatan
         e. Upah perjam
         f. Upah lembur
      1.1.2. Identifikasi Informasi
         a. Laporan Gaji Karyawan
         b. Laporan Data Karyawan
         c. Slip Gaji
2. Identifikasi Sumber Data dan Informasi
   2.1. Identifikasi Sumber Data
      a. Karyawan
      b. Personalia
      c. Direktur
   2.2. Identifikasi Tujuan Informasi
      a. Karyawan
      b. Personalia
      c. Direktur



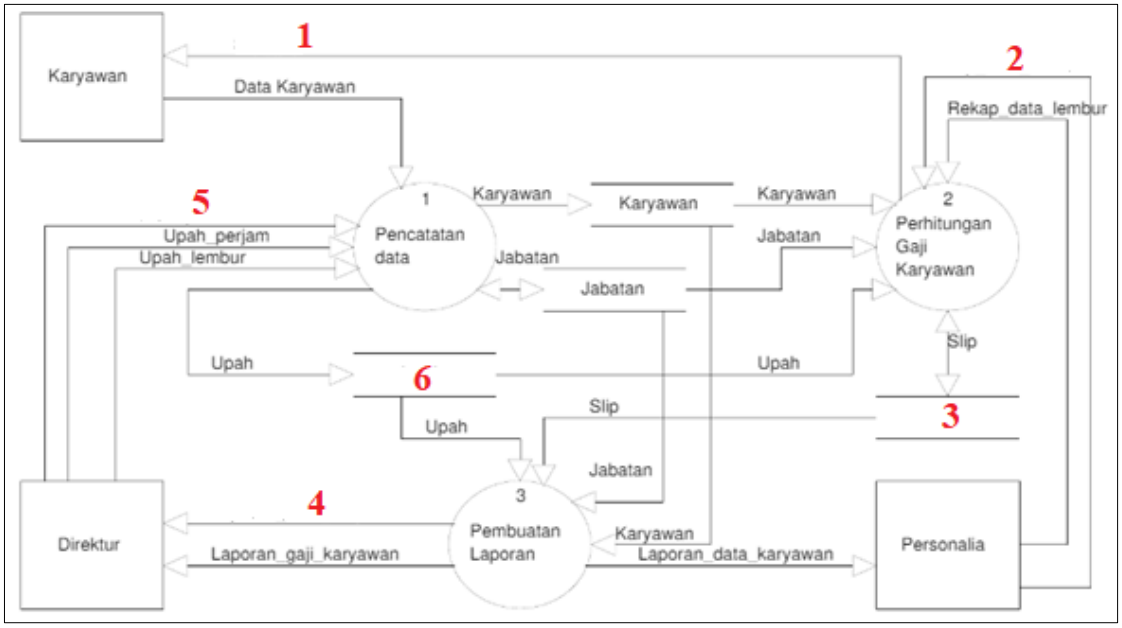
Gambar: Diagram konteks Sistem informasi Penggajian



Gambar: Diagram dekomposisi Sistem informasi Penggajian

Sesi Diskusi

Berdasarkan DFD level 0 Sistem informasi Penggajian dibawah ini lakukan perbaikan dari kekurangan yang ada pada DFD tersebut, berdasarkan keterangan!



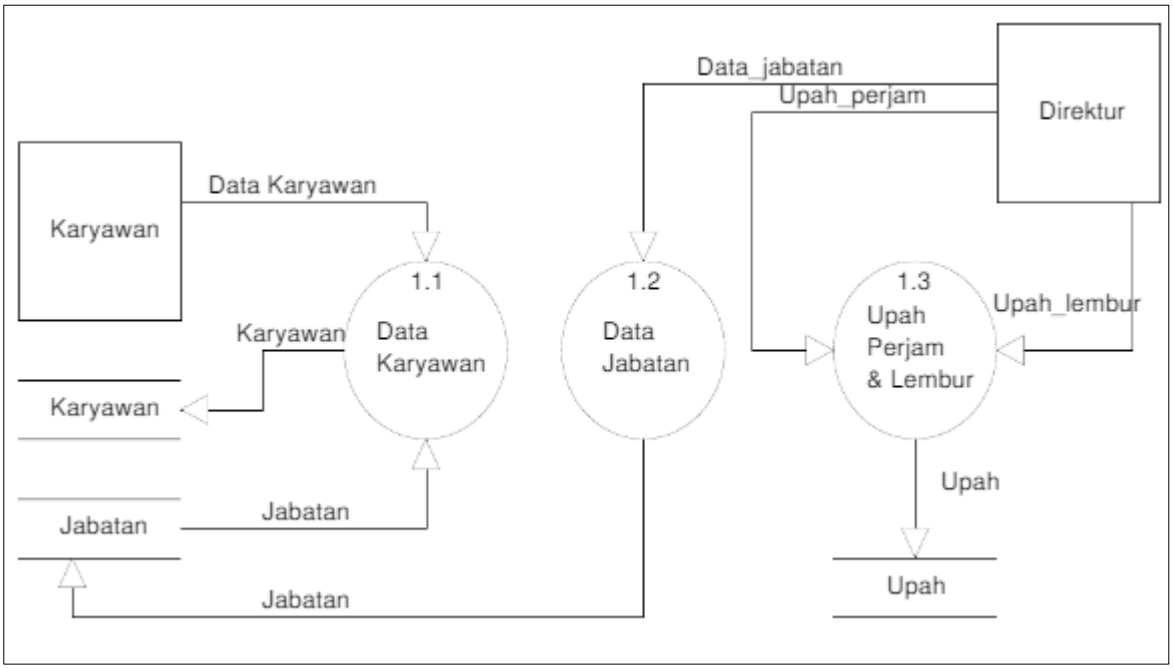
Gambar: DFD Level 0 Sistem informasi Penggajian

Keterangan:

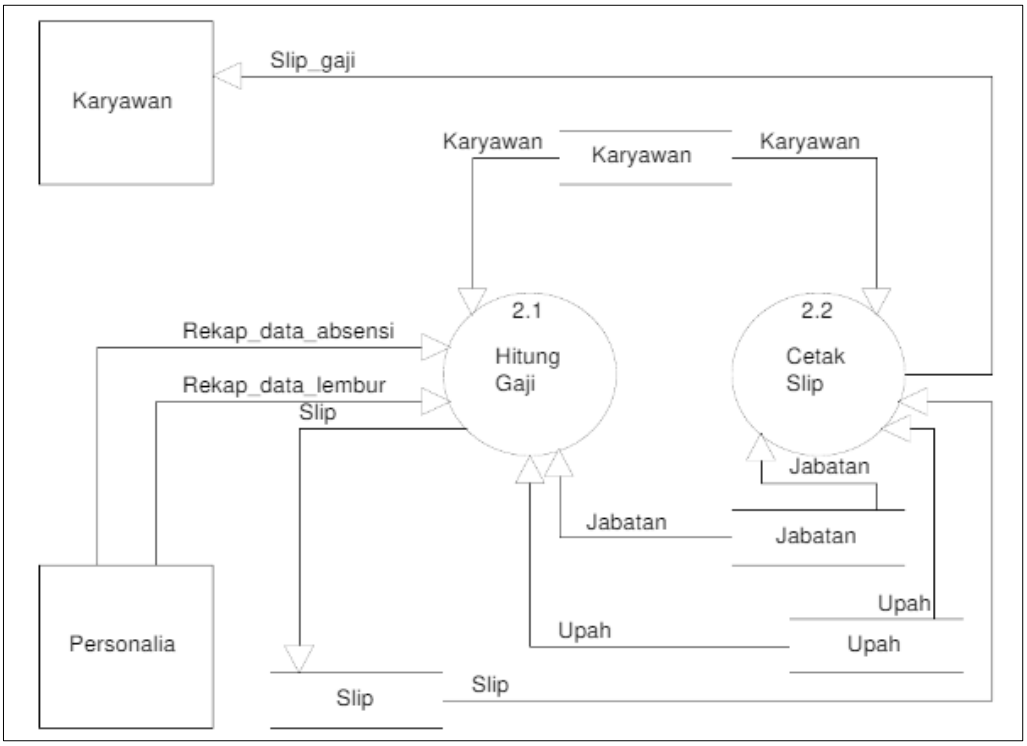
Diagram konteks kemudian diturunkan ke level 0 sehingga proses menjadi 3. Yaitu pencatatan data, perhitungan gaji karyawan dan pembuatan laporan. Pada DFD Level 0, sudah memiliki data store seperti data store karyawan, jabatan, slip dan upah. Rincian proses DFD level 0:

a. Data flow pada entitas eksternal Pegawai memiliki 2 data flow yakni, data karyawan (input) ke proses Pencatatan Data dan slip gaji(ouput) dari proses Perhitungan Gaji Karyawan
b. Data flow pada entitas eksternal Direktur memiliki 5 data flow yakni, Data_jabatan, Upah_perjam, Upah_lembur(input) ke proses Pencatatan Data dan Laporan Gaji Karyawan, Laporan Data Karyawan(output) dari proses pembuatan laporan.
c. Data flow pada entitas eksternal personalia memiliki 3 data flow yakni Rekap_data_lembur, Rekap_data_absensi (input) ke proses Perhitungan gaji karyawan dan Laporan Data Karyawan(output) dari proses pembuatan laporan
d. Pada proses 1 pencatatan data menghasilkan 3 data store dan memiliki 2 data flow serta 1 struktur data. Data flow upah ke data store “upah” dilanjutkan data flow upah ke proses pembuatan laporan. Data flow karyawan ke data store “karyawan” dilanjutkan data flow karyawan ke proses perhitungan gaji karyawan. Terdapat 1 struktur data jabatan ke data store “jabatan” dilanjutkan data flow karyawan ke proses perhitungan gaji karyawan dan ke proses pembuatan laporan.
e. Pada proses 2 perhitungan gaji karyawan menghasilkan 1 data store dan 1 struktur data. Struktur data slip ke data store “slip” dilanjutkan Data flow slip ke proses pembuatan laporan.

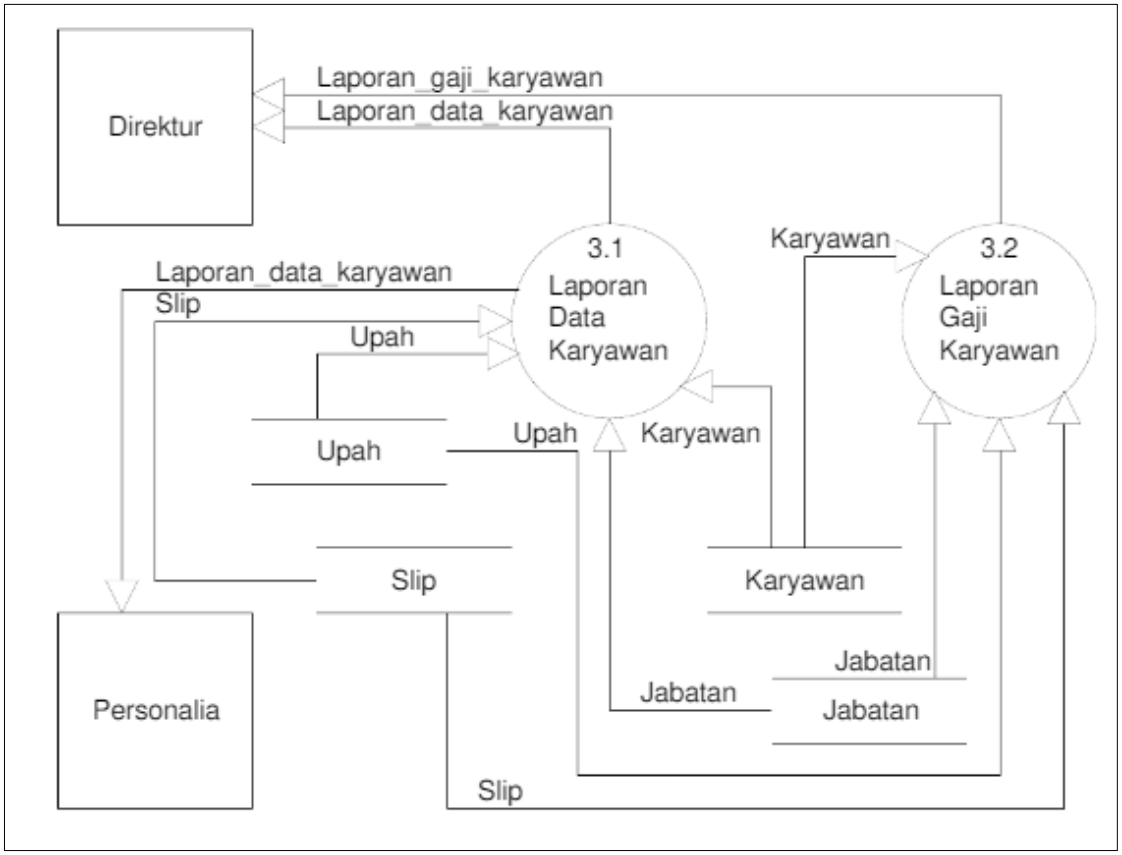
f. Pada proses 3 pembuatan laporan menghasilkan 1 data flow laporan data karyawan ke entitas eksternal personalia.



Gambar: DFD Level 1 proses 1 pencatatan data



Gambar: DFD Level 1 proses 2 perhitungan gaji karyawan



Gambar: DFD Level 1 proses 3 pembuatan laporan

4.4 TUGAS PROGRESS PROJECT AKHIR SEMESTER

Diskusikan bersama kelompok dan buatlah rancangan DFD untuk Perangkat Lunak Anda (melanjutkan tugas sebelumnya).

MODUL 6
Unified Modelling Language (UML)
Use Case Diagram

6.1 Bahasan dan Tujuan

6.1.1 Bahasan

Membahas terkait pemodelan perangkat lunak dengan *Unified modeling Language* (UML) sebagai sarana perancangan sistem berorientasi objek. Dan membahas *use case diagram* sebagai salah satu contoh diagram perancangan pada UML.

6.1.2 Tujuan

1. Mahasiswa memahami pemodelan perangkat lunak dengan pendekatan berorientasi objek
2. Mahasiswa mampu memahami pemodelan *Unified modeling Language* (UML)
3. Mahasiswa mampu memahami dan mendesain *Use Case Diagram* sebagai salah satu pemodelan UML.

6.2 DASAR TEORI

6.2.1 UML

UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek. UML juga dapat didefinisikan sebagai suatu bahasa standar visualisasi, perancangan, dan pendokumentasian sistem, atau dikenal juga sebagai bahasa standar penulisan blueprint sebuah software.

UML diharapkan mampu mempermudah pengembangan piranti lunak (RPL) serta memenuhi semua kebutuhan pengguna dengan efektif, lengkap, dan tepat. Adapun tujuan dan fungsi perlu adanya UML yaitu sebagai berikut:

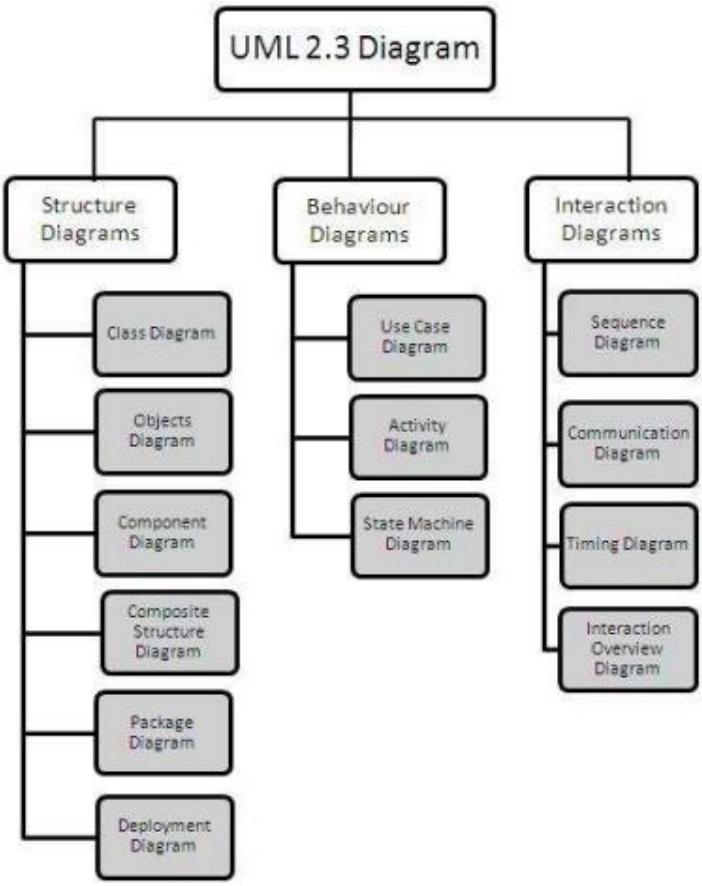
1. Dapat memberikan bahasa pemodelan visual atau gambar kepada para pengguna dari berbagai macam pemrograman maupun proses umum rekayasa.
2. Menyatukan informasi-informasi terbaik yang ada dalam pemodelan.
3. Memberikan suatu gambaran model atau sebagai bahasa pemodelan visual yang ekspresif dalam pengembangan sistem.
4. Tidak hanya menggambarkan model sistem software saja, namun dapat memodelkan sistem berorientasi objek.

5. Mempermudah pengguna untuk membaca suatu sistem.
6. Berguna sebagai blueprint, jelas ini nantinya menjelaskan informasi yang lebih detail dalam perancangan berupa coding suatu program.

UML juga dapat digunakan sebagai alat transfer ilmu tentang sistem aplikasi yang akan dikembangkan dari developer satu ke developer lainnya. UML sangat penting bagi sebagian orang karena UML berfungsi sebagai *bridge* atau jembatan penerjemah antara pengembang sistem dengan pengguna. Di sinilah pengguna dapat memahami sistem yang nantinya akan dikembangkan.

UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut adalah sebagai berikut :

A. **Structure Diagrams,** yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
B. **Behavior Diagrams,** yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakukan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
C. **Interaction Diagrams**, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.



6.2.2 *Use Case Diagram*

Use Case Diagram adalah satu jenis dari diagram UML *(Unified Modelling Language)* yang menggambarkan hubungan interaksi antara sistem dan aktor. Use Case dapat mendeskripsikan tipe interaksi antara si pengguna sistem dengan sistemnya.

A. Fungsi dari use case diagram sebagai berikut:

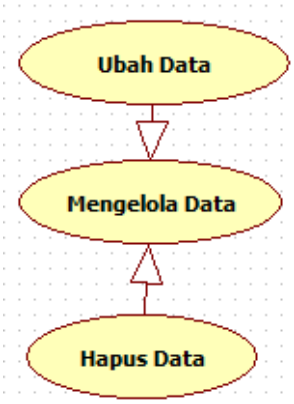
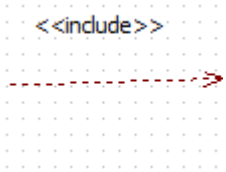
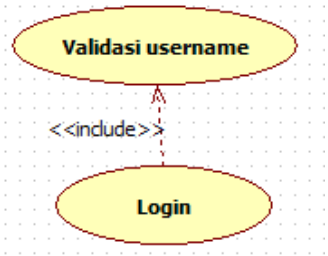
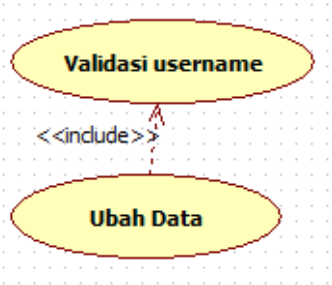
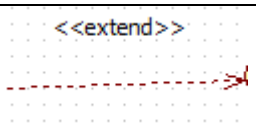
- Berguna memperlihatkan proses aktivitas secara urut dalam sistem.
- Mampu menggambarkan proses bisnis, bahkan menampilkan urutan aktivitas pada sebuah proses.
- Sebagai bridge atau jembatan antara pembuat dengan konsumen untuk mendeskripsikan sebuah sistem.

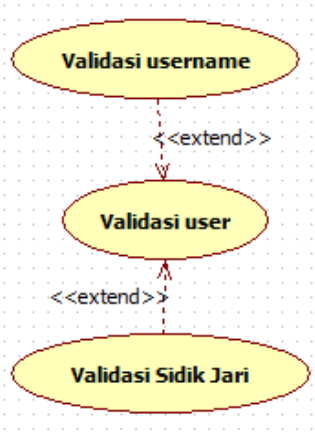
B. Manfaat dari use case di antaranya:

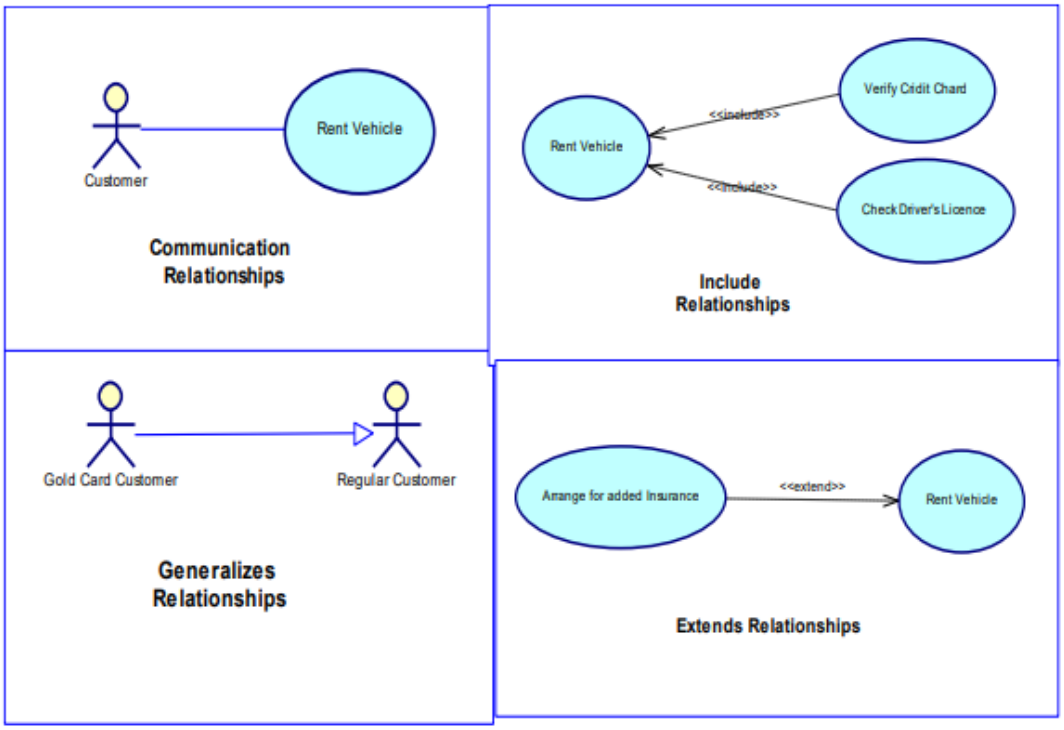
- Menggunakannya sebagai kebutuhan verifikasi.
- Menjadi gambaran interface dari sebuah sistem karena setiap sistem yang dibangun haruslah memiliki interface.
- Mengidentifikasi siapa saja orang yang dapat berinteraksi dengan sistem, serta apa yang dapat dilakukan oleh sistem.
- Memberikan kepastian mengenai kebutuhan sistem, sehingga tidak membingungkan.
- Memudahkan proses komunikasi antara domain expert dan end user.

C. Simbol-simbol pada Use Case Diagram

Simbol	Keterangan
	**Aktor :** Mewakili peran orang, sistem yang lain, ataualat ketika berkomunikasi dengan *use case*
	**Use Case :** Abstraksi dan interaksi antara sistem dan aktor
	**Association :** Abstraksi dari penghubung antara aktor denagan use case

	**Generalization :** Hubungan generalisasi antara 2 buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. 
	Include berarti use case yang ditambahkan akan selalu dipanggil/dilakukan pengecekan saat use case tambahan dijalankan.  
	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan.

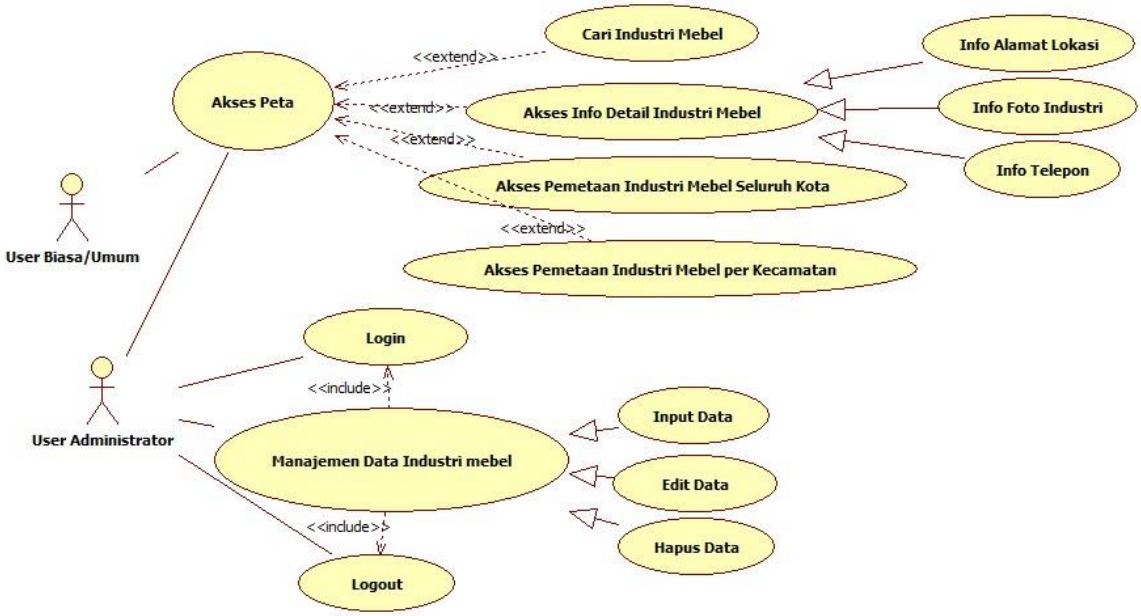
	Relasi ini mirip dengan inheritance pada pemrograman berorientasi objek. 



4.3 PRAKTIKUM

Praktikum 1.

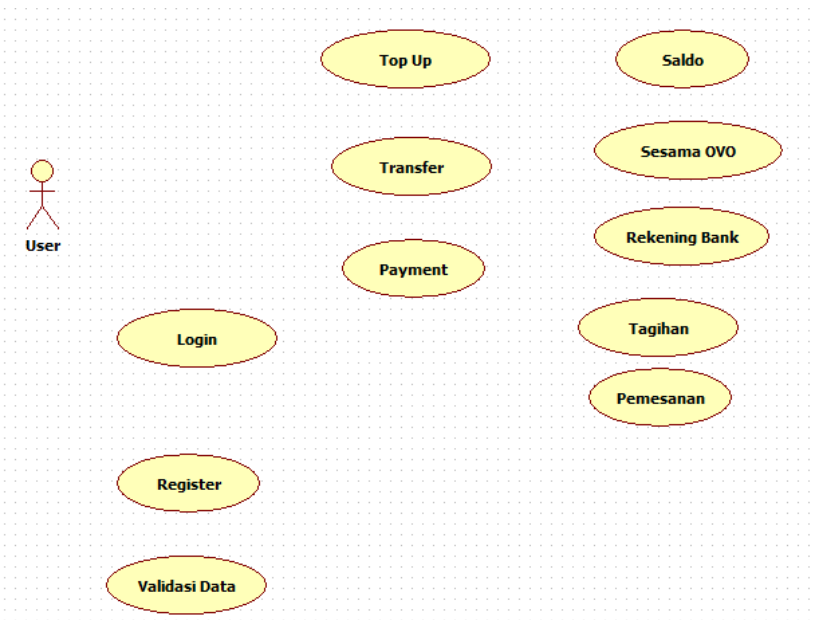
Latihan membuat use case diagram dari Sistem Informasi Geografis Berbasis Website Pemetaan Industri Mebel berikut ini menggunakan tools yang Anda kuasai. Pahami dan analisa alur dan relasi desain sistem yang ada !



Praktikum 2.

Buat dan lengkapilah use case diagram dari analisis perancangan sistem pembayaran OVO berikut ini.

OVO adalah aplikasi pintar yang memberikan kemudahan dalam bertransaksi (OVO Cash), serta berfungsi untuk mengumpulkan poin di banyak tempat (OVO Points). Lengkapi use case diagram studi kasus pada OVO berikut ini.



- ***User*****:** Orang yang dapat mengakses atau menggunakan aplikasi OVO, mulai dari login ke aplikasi hingga melakukan aksi terhadap aplikasi seperti top up saldo, transfer, dan payment.
- ***Register*****:** Register merupakan langkah pertama yang dilakukan user ketika ia tidak mempunyai akses pada aplikasi OVO. Mendaftarkan data diri ke dalam aplikasi agar dikenali.
- **Login:** Setelah mendapatkan akun, user harus melakukan login agar dapat mengakses berbagai fitur aplikasi OVO.
- ***Top up*****:** Suatu kegiatan yang dilakukan user untuk mengisi ulang saldo OVO. Terdapat 2 pilihan alternatif untuk melakukan top up saldo, yaitu melalui ATM dan internet banking.
- ***Transfer*****:** Transfer berfungsi untuk mengirim atau membagikan saldo dalam aplikasi OVO ke pengguna lain, baik sesama OVO atau ke rekening tertentu.
- ***Payment*****:** Ketika user memilih menu payment, maka user dapat melakukan pembayaran lewat aplikasi.

6.4 TUGAS PROGRESS PROJECT AKHIR SEMESTER

Diskusikan bersama kelompok dan buatlah rancangan *Use Case Diagram* untuk Perangkat Lunak Anda (melanjutkan tugas sebelumnya).

MODUL 7
Unified Modelling Language (UML)
Class Diagram

7.1 Bahasan dan Tujuan

7.1.1 Bahasan

Membahas terkait dengan salah satu pemodelan yang ada di dalam *Unified modeling Language* (UML) yaitu *Class Diagram*.

7.1.2 Tujuan

1. Mahasiswa memahami pemodelan perangkat lunak dengan pendekatan berorientasi objek.
2. Mahasiswa mampu memahami pemodelan *Unified modeling Language* (UML).
3. Mahasiswa mampu memahami dan mendesain *Class Diagram* sebagai salah satu pemodelan UML.

7.2 DASAR TEORI

7.2.1 *Structure Diagrams*

Mengulang kembali materi yang ada pada modul sebelumnya, UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek. UML juga dapat didefinisikan sebagai suatu bahasa standar visualisasi, perancangan, dan pendokumentasian sistem, atau dikenal juga sebagai bahasa standar penulisan *blueprint* sebuah *software*.

UML terdiri dari 13 macam *diagram* yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut salah satunya adalah *Structure Diagrams***.** *Structure Diagrams* adalah kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

7.2.2 *Class Diagram*

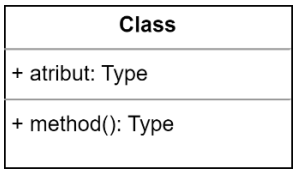
Class Diagram atau diagram kelas adalah satu jenis dari *Structure Diagrams* yang ada pada UML *(Unified Modelling Language*). *Class Diagram* digunakan untuk menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* bersifat statis, dalam artian diagram kelas bukan menjelaskan apa yang terjadi jika kelas-kelasnya berhubungan, melainkan menjelaskan hubungan apa yang terjadi.

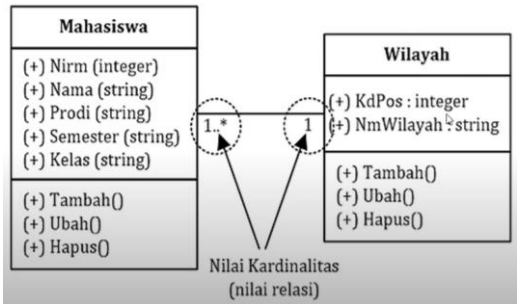
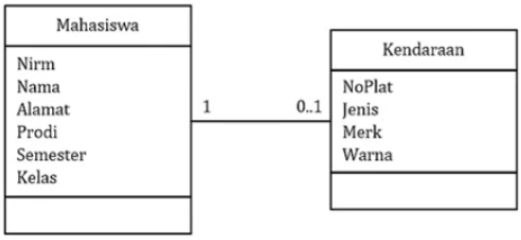
Class Diagram dibuat dengan tujuan agar *programmer* membuat kelas-kelas sesuai rancangan di dalam *Class Diagram* sehingga antara dokumentasi perancangan dan perangkat lunak yang dibangun sinkron.

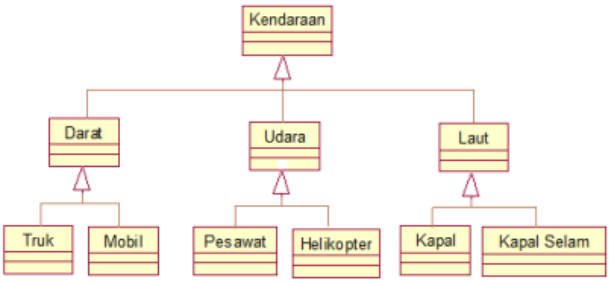
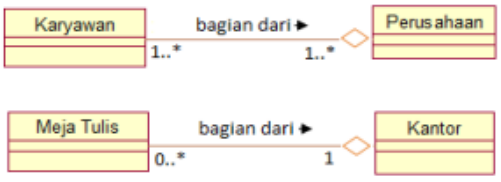
Kelas-kelas yang ada pada struktur sistem harus mampu melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga *programmer* dapat membuat kelas-kelas di dalam perangkat lunak sesuai dengan perancangan *Class Diagram*. Susunan struktur *class* yang baik pada *Class Diagram* memiliki jenis-jenis kelas sebagai berikut:

a. Kelas main (*Main*), kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b. Kelas yang menampilkan tampilan sistem (*View*), kelas yang mendefinisikan dan mengatur tampilan ke pengguna.
c. Kelas yang diambil dari pendefinisian *use case* (*Controller*), kelas yang menangani fungsi-fungsi yang harus ada. *Controller class* biasa disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
d. Kelas yang diambil dari pendefinisian data (*Model*), kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan pada basis data.

Simbol-simbol pada *Class Diagram*

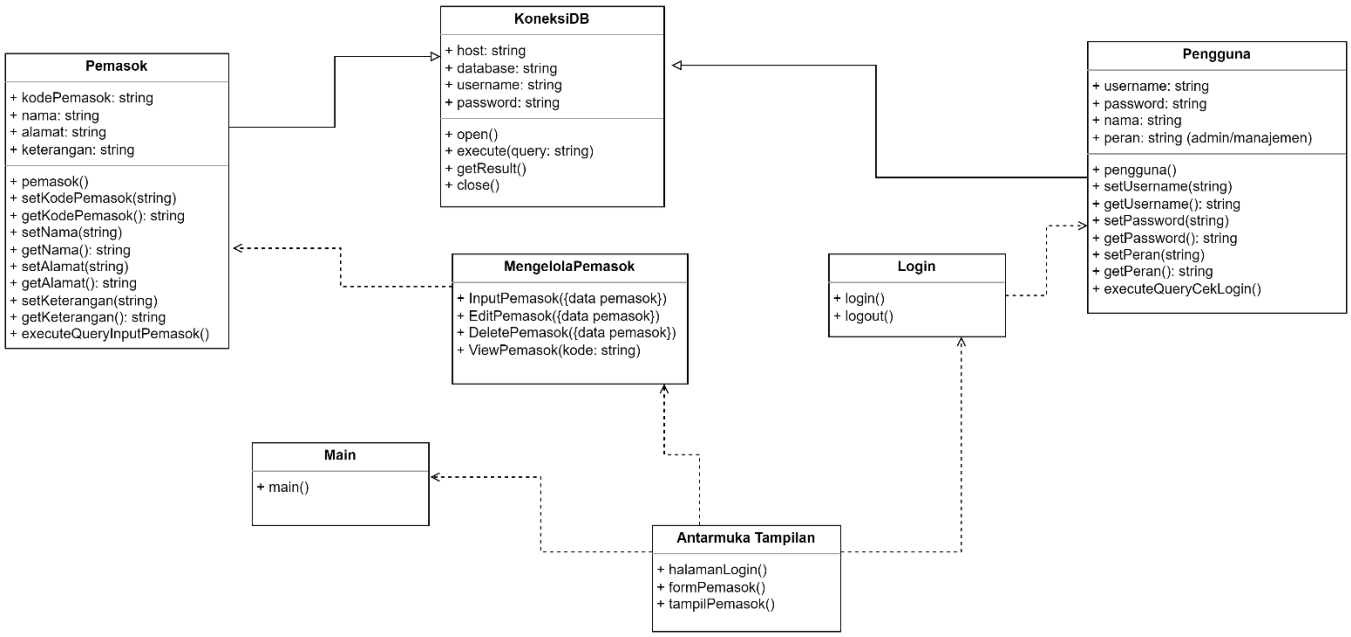
Simbol	Keterangan
	***Class*** • Menggambarkan *class* pada struktur sistem. • *Class* digambarkan dengan persegi panjang dengan tiga baris: nama *class*, atribut *class*, dan metode atau operasi yang mungkin digunakan *class*. • Memiliki nama yang dicetak tebal dan berada di tengah kompartemen atas. ***Attribute*** • Notasi atribut mendeskripsikan properti dengan sebaris teks di dalam kotak *class*. • Aturan penulisan atribut: `visibility name: type multiplicity = default (property-string)` *Visibility*: menandakan atribut public (+), private (-), dan protected (#) *Name*: nama atribut

	Type: batasan tentang objek yang dapat diletakkan dalam *Atribut default value*: nilai dari objek *(property-string)*: properti tambahan. ***Method*** Merepresentasikan tindakan atau fungsi yang dapat dilakukan oleh *class*.
_____	***Association* (asosiasi)** • Merepresentasikan hubungan antara beberapa kelas, atau kelas dan dirinya sendiri. • Dapat dilabeli dengan frase kata kerja atau nama peran, atau apa saja yang lebih mewakili hubungan. • Biasanya disertai dengan *multiplicity*. ***Multiplicity*** • Merupakan indikasi tentang berapa banyak objek yang bisa berhubungan dengan objek lain. • Angka ditempatkan pada jalur asosiasi untuk menunjukkan jumlah minimum dan maksimum yang dapat dihubungkan melalui asosiasi dalam format angka minimum angka maksimum. • Bentuk *multiplicity*: **1** : *exactly one* **0 …. *** : *zero or more* **1 …. *** : *one or more* **0 …. 1** : *zero or one*  

	***Directed Association* (asosiasi berarah):** • Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain. • Biasanya disertai dengan *multiplicity*.
	***Generalization* (generalisasi):** • Relasi antar kelas dengan makna generalisasi-spesialisasi (umum ke khusus). • Contoh: 
	***Dependency* (kebergantungan):** Relasi antar kelas dengan makna kebergantungan antar kelas.
	***Aggregation* (agregasi):** • Agregasi digunakan ketika kelas sebenarnya terdiri dari kelas lain. • Asosiasi agregasi biasanya diidentifikasi ketika kita perlu menggunakan kata-kata seperti "adalah bagian dari" atau "terdiri dari" untuk menggambarkan hubungan. • Contoh: 

Catatan:
Arah panah relasi pada *class diagram* mengarah pada *class diagram* yang lebih besar kontrolnya atau yang dipakai.

Contoh *Class Diagram*

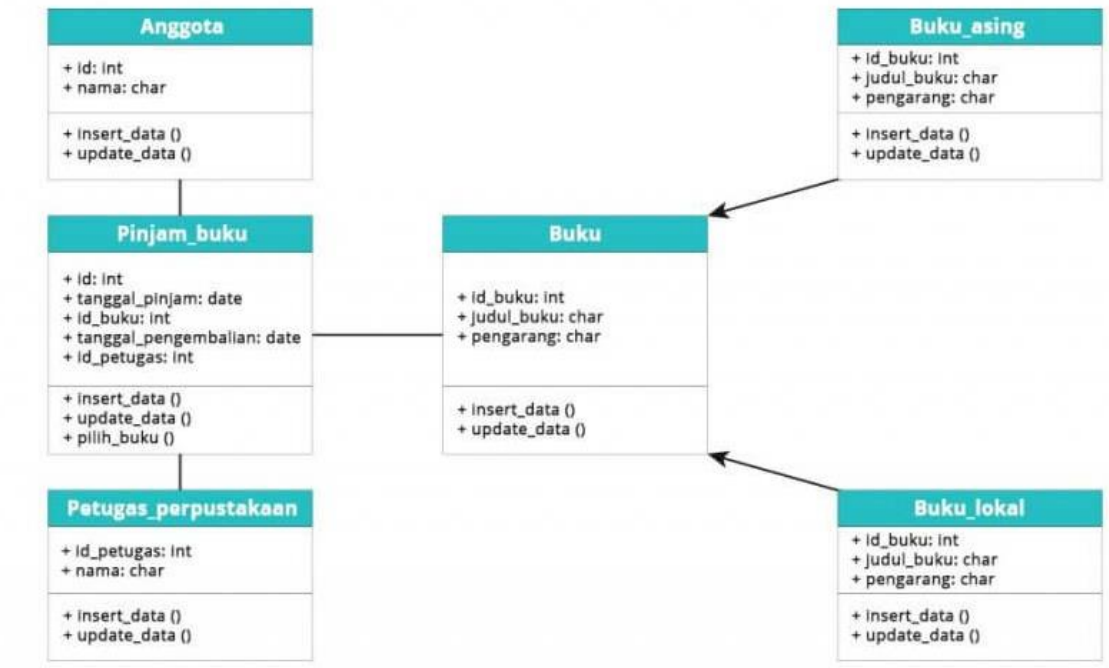


7.3 PRAKTIKUM

Buat ulang (*recreate*) dan pahami *class diagram* di bawah ini.

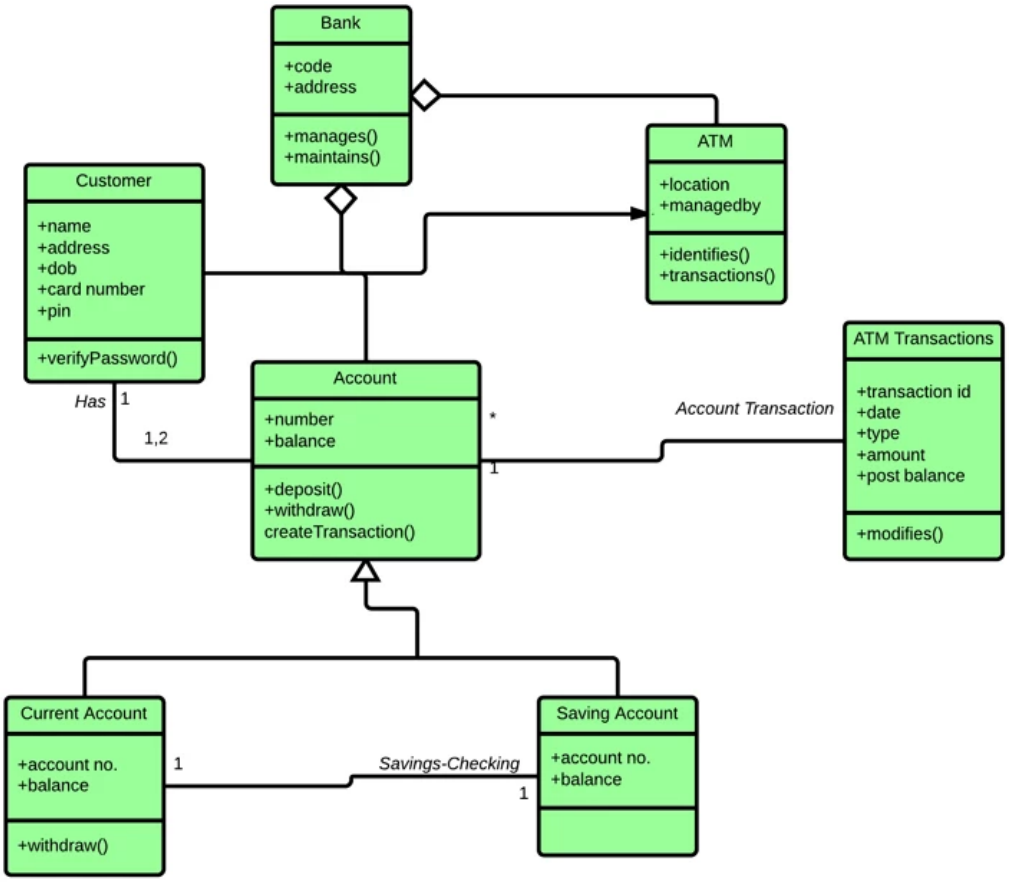
- *Class Diagram* Perpustakaan

  Jelaskan maksut *class diagram* di bawah ini.



- *Class Diagram* Mesin ATM

Jelaskan maksut *class diagram* di bawah ini.



7.4 TUGAS PROGRESS PROJECT AKHIR SEMESTER

Diskusikan bersama kelompok dan buatlah rancangan *Class Diagram* untuk Perangkat Lunak Anda (melanjutkan tugas sebelumnya).

MODUL 8

Sequence Diagram

1.1 Bahasan dan Tujuan

1.1.1 Bahasan

Mampu menjelaskan pemodelan dan diagram UML serta merancang model UML dengan tools yang sesuai.

1.1.2 Tujuan

1. Praktikan mengenal tentang *Sequence Diagram*
2. Praktikan mengenal tentang Model Perilaku berbasis *Sequence Diagram*
3. Praktikan mengimplementasikan *Sequence Diagram* pada studi kasus

1.2 DASAR TEORI

1.2.1. Pengertian Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri antara dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Dalam Sequence Diagram, setiap object hanya memiliki garis yang digambarkan garis putus-putus ke bawah. Pesan antar object digambarkan dengan anak panah dari object yang mengirimkan pesan ke object yang menerima pesan.

Sequence Diagram dapat berupa Sequence Diagram umum yang menunjukkan semua kemungkinan skenario untuk use case, atau analis hanya mengembangkan satu set Sequence Diagram pada lingkup instance, yang masing-masing menggambarkan skenario tunggal dalam use case.

1.2.2. Fungsi Sequence Diagram:

a. Menentukan detail dari Use Case.
b. Memodelkan logika prosedur, fungsi, atau operasi yang terdapat dalam sistem.
c. Untuk melihat bagaimana objek dan komponen saling berinteraksi satu sama lain untuk menyelesaikan suatu proses.
d. Merencanakan dan memahami fungsionalitas secara rinci dari skenario yang ada atau yang akan datang.

1.2.3. Bagian-Bagian Sequence Diagram

A. Tabel 1: Simbol Sequence Diagram.

No	Simbol	Nama	Fungsi
1	:Objek	Object	Komponen utama Sequence Diagram
2		Actor	Menggambarkan orang yang sedang berinteaksi dengan sistem
3		Entity Class	Menggambarkan hubungan kegiatan yang akan dilakukan
4		Boundary Class	Menggambarkan sebuah penggambaran dari form
5		Control Class	Menggambarkan penghubung antara boundary dengan tabel
6		Life Line	Menggambarkan tempat mulai dan berakhirnya sebuah message
7		Message	Menggambarkan pengiriman pesan

Sequence Diagram yang menggambarkan interaksi objek/kelas yang menyertakan objek database atau komponen GUI (*Graphical User Interface*) tertentu sebagai kelas pada sebuah program aplikasi (*software*). Kelas-kelas biasanya diidentifikasi berdasarkan jenisnya, yaitu: *Boundary Class, Control Class,* dan *Entity Class* seperti pada tabel 1. Boundary Class adalah kelas (class) yang menghubungkan user dengan sistem aplikasi. Oleh karena itu sering disebut sebagai user interface class. Control Class adalah kelas yang mengkoordinasi atau mengendalikan aktivitas dalam sistem. Kelas ini menghubungkan Boundary Class dengan Entity Class. Entity Class adalah kelas yang berhubungan dengan data atau informasi yang digunakan oleh sistem, yaitu media penyimpanan data yang dapat berupa sebuah database/ tabel (Hermawan, 2004).

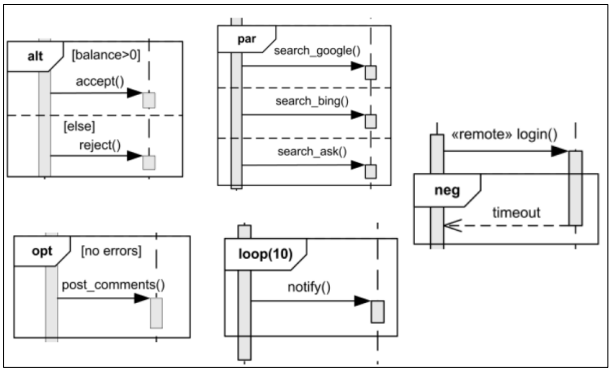
B. Loop dan Kondisi

1. Loop dan kondisi menggunakan kerangka interaksi, yaitu cara penandaan sebuah bagian sequence diagram.
2. Kerangka terdiri dari beberapa daerah yang dipisahkan menjadi beberapa fragmen.
3. Setiap kerangka memiliki sebuah operator.
4. Setiap fragmen memiliki sebuah guard.
5. Guard merupakan sebuah ekspresi kondisional dalam tanda kurung [ ], dan menunjukkan bahwa pesan akan dikirimkan jika nilai guard benar.

C. Operator Umum untuk Kerangka Interaksi

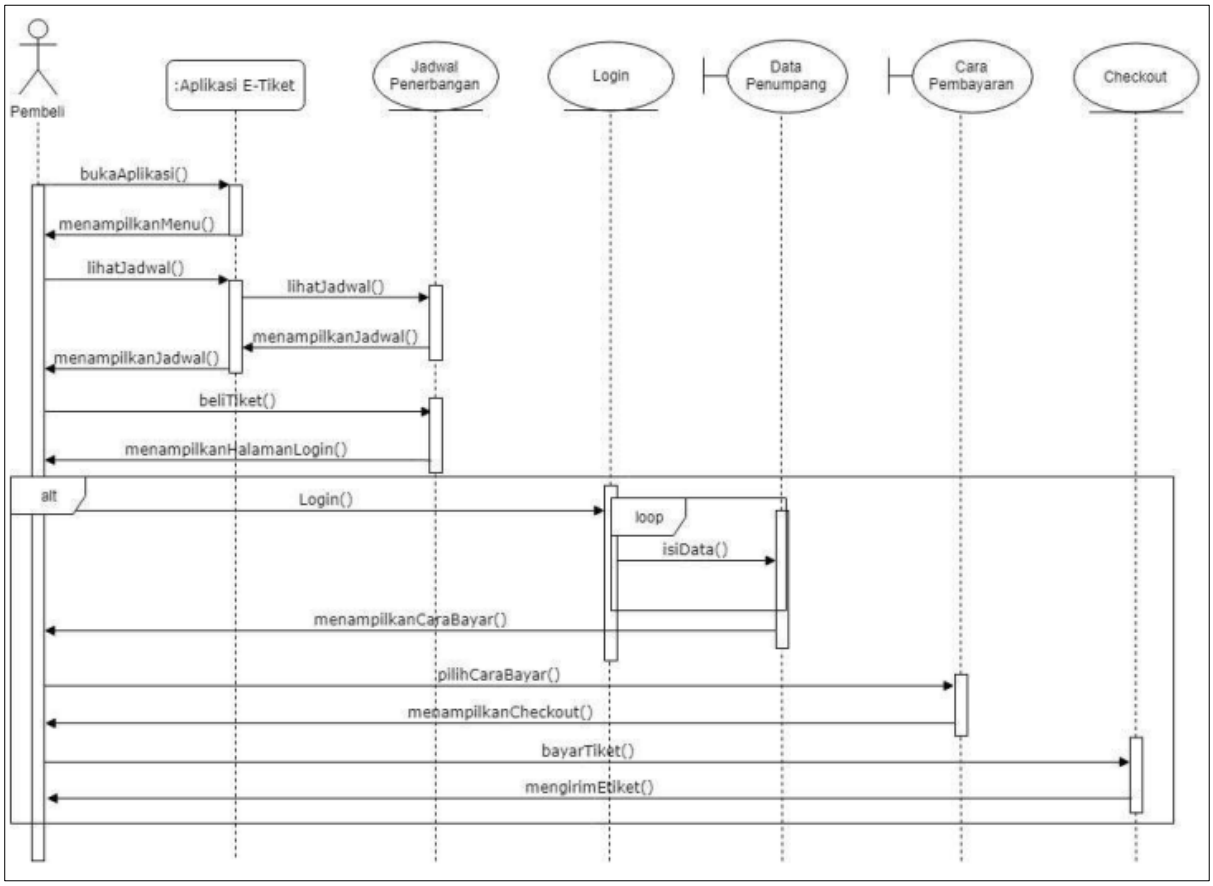
Tabel: Operator

Operator	Keterangan
alt	Alternatif dari banyak fragmen. **Hanya yang kondisinya true yang akan dijalankan**
opt	Optional; fragmen akan dijalankan **jika kondisi yang mendukungnya true**
par	Paralel; setiap fragmen dijalankan secara paralel
loop	Looping, fragmen mungkin dijalankan berulang kali dan guard menunjukkan basis iterasi
region	Critical region; fragmen hanya dapat mempunyai satu thread untuk menjalankannya
neg	Negatif; fragmen menunjukkan interaction yang salah
ref	Reference; menunjukkan ke sebuah interaction yang didefinisikan pada diagram yang lain
sd	Sequence diagram



Gambar: Contoh penggunaan operator

1.3 PRAKTIKUM



Gambar: Contoh Sequence Diagram Aplikasi E-tiket

Keterangan:

1. Sistem memiliki 1 aktor yaitu pembeli
2. Memiliki 1 object yaitu aplikasi E-Tiket Pesawat
3. Terdapat 5 kelas yang saling berinteraksi, yaitu jadwal penerbangan, login, data penumpang, cara pembayaran dan checkout.
4. Pembeli dapat membeli tiket jika login
5. Detail proses:
   a. Pembeli membuka object “aplikasi E-Tiket”.
   b. Object “Aplikasi E-Tiket” menampilkan Menu ke pembeli.
   c. Pembeli dapat melihat jadwal pada object “aplikasi E-Tiket”, object “aplikasi E-Tiket” berinteraksi dengan class “jadwal penerbangan” untuk informasi lihat jadwal. class “jadwal penerbangan” menampilkan jadwal ke object “aplikasi E-Tiket”, untuk menampilkan jadwal ke pembeli.
   d. Object “Aplikasi E-Tiket” menampilkan jadwal ke pembeli.
   e. Terdapat fragmen “alt” mulai dari proses login hingga mengirim e-tiket.
   f. Pembeli beli tiket pada class “login”, terdapat fragmen “loop” pada proses isi data. class “login” meneruskan isi data ke class “data penumpang”.
   g. Class “data penumpang” menampilkan cara membayar ke pembeli.
   h. Pembeli memilih cara membayar ke class “cara pembayaran”, class “data pembayaran” menampilkan checkout ke pembeli.
   i. Pembeli membayar tiket ke class “checkout”, class “checkout” mengirim e-tiket ke pembeli.

Tugas.

Buatlah Sequence Diagram yang menjelaskan setiap skenario yang ada dalam project UAS beserta keterangan!

MODUL 9
Pengujian Perangkat Lunak

6.1 Bahasan dan Tujuan

6.1.1 Bahasan

Membahas tahapan pada proses pengujian perangkat lunak dan beberapa macam contoh metode pada pengujian perangkat lunak.

6.1.2 Tujuan

1. Mahasiswa memahami proses pengujian perangkat lunak
2. Mahasiswa mampu merancang kasus uji
3. Mahasiswa mampu memahami dan melakukan pengujian *black box testing* dan *white box testing*

6.2 DASAR TEORI

6.2.1 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya. Kualitas suatu perangkat lunak sangat bergantung pada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan sebagai berikut :

a. Agar dapat "*survive*" bertahan hidup di dunia bisnis perangkat lunak.
b. Dapat bersaing dengan perangkat lunak yang lain.
c. Penting untuk pemasaran global (*global marketing*).
d. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran produksi.
e. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak berada di tangan user. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan "***bug".*** Untuk menghindari banyaknya bug maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan.

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi dan validasi / *verification and validation* (V&V).

a) **Verifikasi :** Sekumpulan aktifitas yang menjamin bahwa software telah diimplentasikan dengan benar untuk suatu fungsi tertentu **(*Are we building the product right ?*)**

b) **Validasi :** Sekumpulan aktifitas yang menjamin bahwa suatu software yang dibuat telah sesuai dengan kebutuhan user **(Are we building the right product?)**

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

1. *Black-Box Testing* (Pengujian kotak hitam)
2. *White-Box Testing* (Pengujian kotak putih)

6.2.2 Black-Box Testing

Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan.

6.2.3 *White-Box Testing*

Pengujian kotak putih dilakukan dengan memeriksa lojik (*logical path*) dari kode program (*source code*) dan desain program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang seharusnya.

Contoh dari *white-box testing* misalkan menguji alur Loop (*Looping Testing*), alur Conditions (*Condition Branch*).

Keuntungan:

Menghasilkan program yang benar dan sempurna 100%,

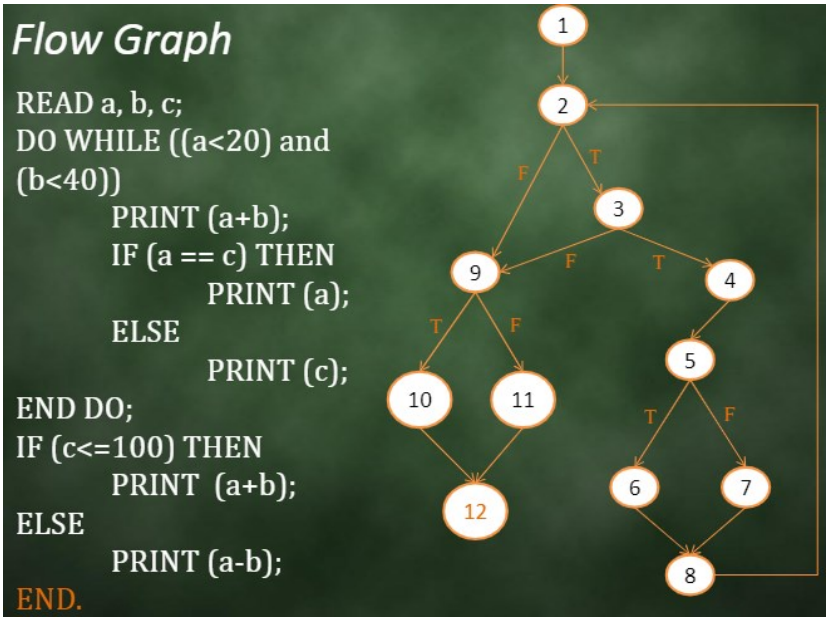
karena:

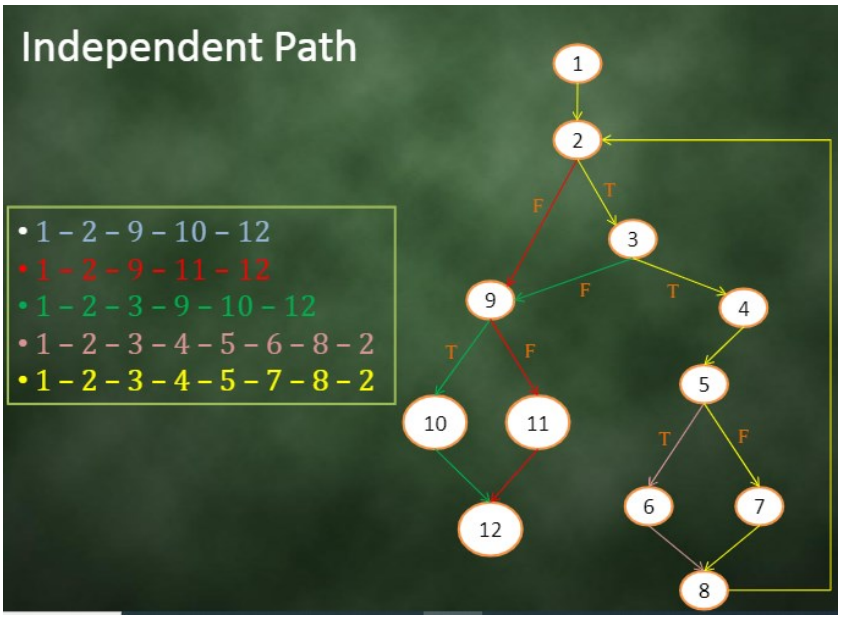
- Mengerjakan seluruh keputusan logika
- Mengerjakan seluruh loop (sesuai batas)
- Menjamin seluruh jalur independen dalam modul dikerjakan minimal 1x
- Mengerjakan seluruh data internal yang menjamin validitas

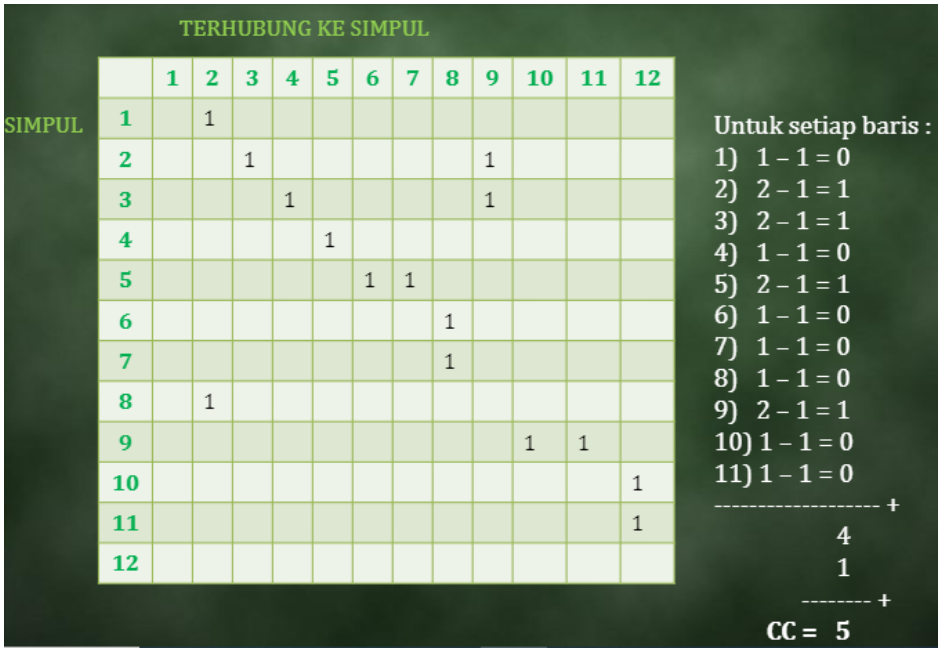
Kekurangan:

- Pengujian secara menyeluruh justru menimbulkan masalah sumber daya
- Program yang kecil bisa menghasilkan banyak sekali jalur logika

Pengujian White-box salah satunya menggunakan teknik **Basis Path Testing** (Berbasis Alur), dimana tujuannya meyakinkan bahwa test case akan menguji setiap path pada suatu program paling sedikit satu kali. Pengujian ini memungkinkan perancangan kasus uji yang diturunkan dari pengukuran tingkat kompleksitas (*cyclomatic complexity*) struktur program. Ukuran tingkat kompleksitas tersebut menjadi panduan di dalam menentukan jalur-jalur dasar (*basis path*). Kasus uji yang didapat menjadi kasus uji yang digunakan untuk membuktikan bahwa setiap pernyataan (*statement*) dari program akan dieksekusi minimal sekali.

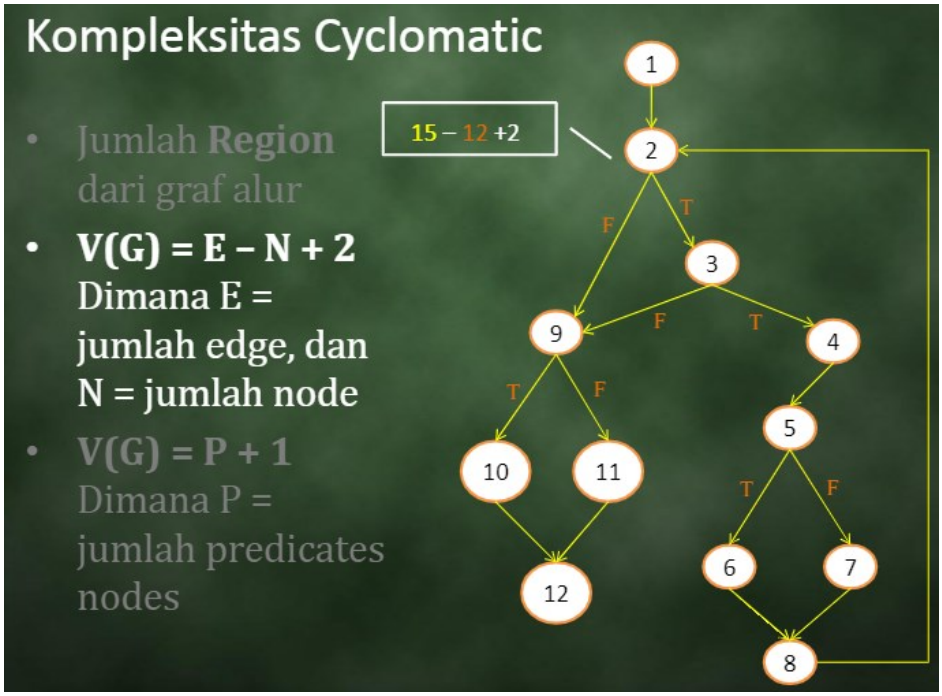






	1	2	3	4	5	6	7	8	9	10	11	12
1		1										
2			1						1			
3				1					1			
4					1							
5						1	1					
6								1				
7								1				
8		1										
9										1	1	
10												1
11												1
12												

Atau menghitung CC (*Cyclomatic Complexity*) dengan rumus V(G) = E-N+2.



V(G) = E-N+2 = 15-12+2 = 5.

4.3 PRAKTIKUM

Contoh

Black-box testing

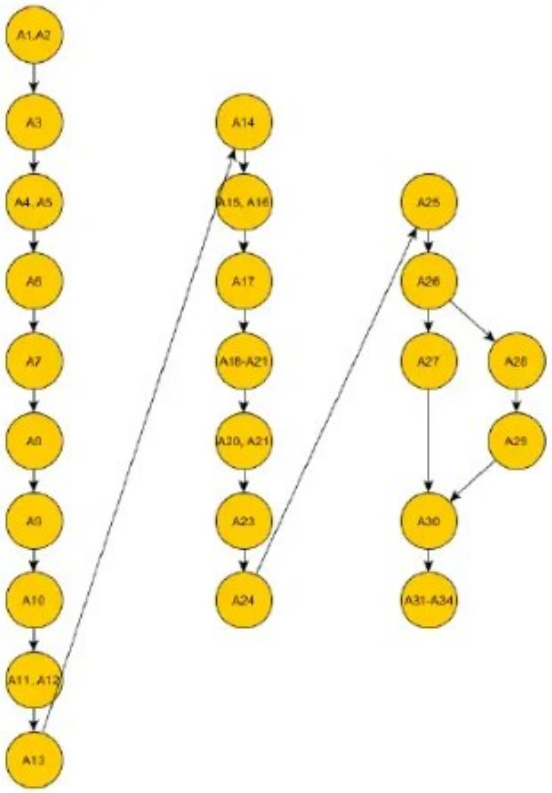
Kasus uji (*test case*) yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login Admin maka kasus uji yang dibuat adalah:

No.	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	*Username* dan *Password* tidak diisi kemudian klik tombol Login	Username: (kosong) Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Harap isi username dan password"	Sesuai harapan	Vaid
2	Mengetikkan Username, dan password tidak diisi atau kosong kemudian klik tombol Login	Username: admin Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Password belum diisi"	Sesuai harapan	Vaid
3	Mengetikkan Password, dan username tidak diisi atau kosong kemudian klik tombol Login	Username: (kosong) Password: admin	Sistem akan menolak dan menampilkan pesan "Username belum diisi"	Sesuai harapan	Vaid
4	Mengetikkan Username dan/atau password tidak sesuai, kemudian klik tombol Login	Username: adm Password: adm123	Sistem akan menolak dan menampilkan pesan "Username atau Password yang anda masukan salah"	Sesuai harapan	Vaid
5	Mengetikkan Username dan password (diisi), kemudian klik tombol Login	Username: admin Password: admin	Sistem menerima akses login dan kemudian menampilkan halaman utama Admin	Sesuai harapan	Vaid

Contoh
White-box testing

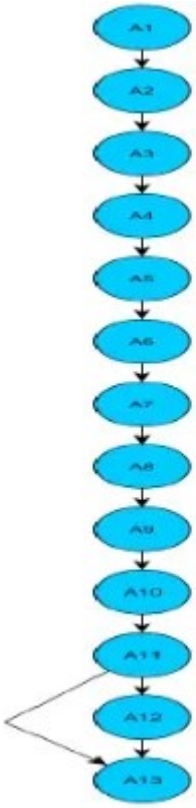
Source code: Index.php

```
1.  <html>
2.  <head>
3.  <link rel="stylesheet" type="text/css" href="style.css">
4.  </head>
5.  <body>
6.  <div class="login">
7.  <form action="login.php" method="post" onSubmit="return validasi()">
8.  <div>
9.  <label>Username:</label>
10. <input type="text" name="username" id="username" />
11. </div>
12.
13. <label>Password:</label>
14. <input type="password" name="password" id="password" />
15. </div>
16. <div>
17. <input type="submit" value="Login" class="tombol">
18. </div>
19. </form>
20. </div>
21. </body>
22. <script type="text/javascript">
23. function validasi() {
24. var username = document.getElementById("username").value;
25. var password = document.getElementById("password").value;
26. if (username != "" && password!="") {
27. return true;
28. }else{
29. alert('Username dan Password harus di isi !');
30. return false;
31. }
32. }
33. </script>
34. </html>
```



Source code: Login.php

```
1.  <?php
2.  include 'config.php';
3.  $username = $_POST['username'];
4.  $password = md5($_POST['password']);
5.  $login = mysql_query("select * from user where
    username='$username' and password='$password'");
6.  $cek = mysql_num_rows($login);
7.  session_start()if($cek > 0){
8.  $_SESSION['username'] = $username;
9.  $_SESSION['status'] = "login";
10. header("location:admin/index.php");
11. }else{
12. header("location:index.php");
13. }?>
```



Selanjutnya analisa hasil halaman pada aplikasi apakah sudah benar dan berfungsi dengan baik.

Praktikum : Lakukan studi literatur melalui beberapa jurnal yang mengimplementasikan masing-masing pendekatan testing tersebut untuk lebih memahami proses pengujian pada kedua metode.

6.3 TUGAS PROGRESS PROJECT AKHIR SEMESTER

1. Diskusikan bersama kelompok untuk studi literatur melalui jurnal, yang mengimplementasikan *Black-Box Testing* (Pengujian kotak hitam) dan *White-Box Testing* (Pengujian kotak putih).

Masing- masing 1 jurnal, dengan ketentuan jurnal berbeda dengan kelompok yang lain. Jika sama nilai akan dibagi.

2. Jurnal tersebut di upload pada G-drive berikut paling lambat 24 Mei 2023. Link Gdrive: https://drive.google.com/drive/folders/14RNt5ZcGDU2GLK6Ne2pgOlj52rse7df7?usp=share_link

MODUL 9
Pengujian Perangkat Lunak

6.1 Bahasan dan Tujuan

6.1.1 Bahasan

Membahas tahapan pada proses pengujian perangkat lunak dan beberapa macam contoh metode pada pengujian perangkat lunak.

6.1.2 Tujuan

1. Mahasiswa memahami proses pengujian perangkat lunak
2. Mahasiswa mampu merancang kasus uji
3. Mahasiswa mampu memahami dan melakukan pengujian *black box testing* dan *white box testing*

6.2 DASAR TEORI

6.2.1 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya. Kualitas suatu perangkat lunak sangat bergantung pada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan sebagai berikut :

a. Agar dapat "*survive*" bertahan hidup di dunia bisnis perangkat lunak.
b. Dapat bersaing dengan perangkat lunak yang lain.
c. Penting untuk pemasaran global (*global marketing*).
d. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran produksi.
e. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak berada di tangan user. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan "***bug".*** Untuk menghindari banyaknya bug maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan.

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi dan validasi / *verification and validation* (V&V).

a) **Verifikasi :** Sekumpulan aktifitas yang menjamin bahwa software telah diimplentasikan dengan benar untuk suatu fungsi tertentu **(*Are we building the product right ?*)**

b) **Validasi :** Sekumpulan aktifitas yang menjamin bahwa suatu software yang dibuat telah sesuai dengan kebutuhan user **(Are we building the right product?)**

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

1. *Black-Box Testing* (Pengujian kotak hitam)
2. *White-Box Testing* (Pengujian kotak putih)

6.2.2 Black-Box Testing

Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan.

6.2.3 *White-Box Testing*

Pengujian kotak putih dilakukan dengan memeriksa lojik (*logical path*) dari kode program (*source code*) dan desain program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang seharusnya.

Contoh dari *white-box testing* misalkan menguji alur Loop (*Looping Testing*), alur Conditions (*Condition Branch*).

Keuntungan:

Menghasilkan program yang benar dan sempurna 100%,

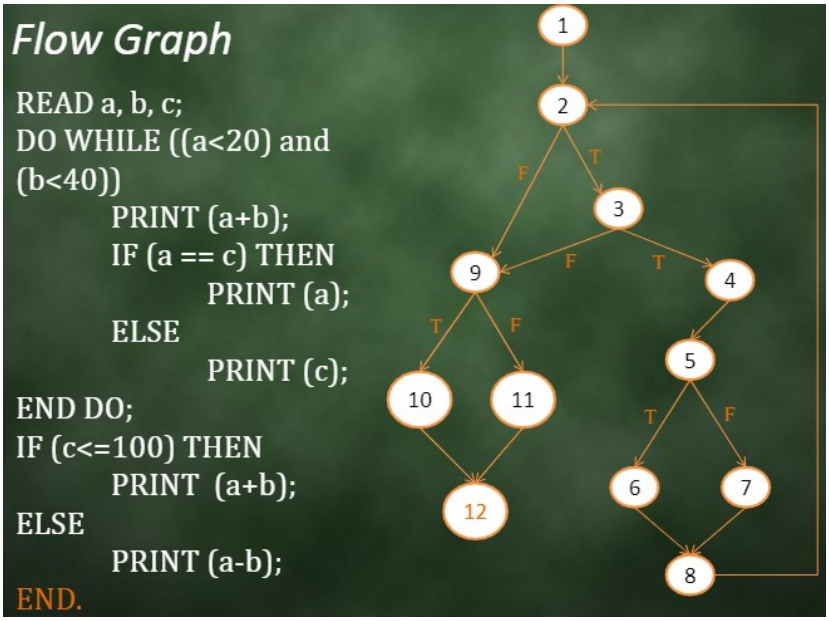
karena:

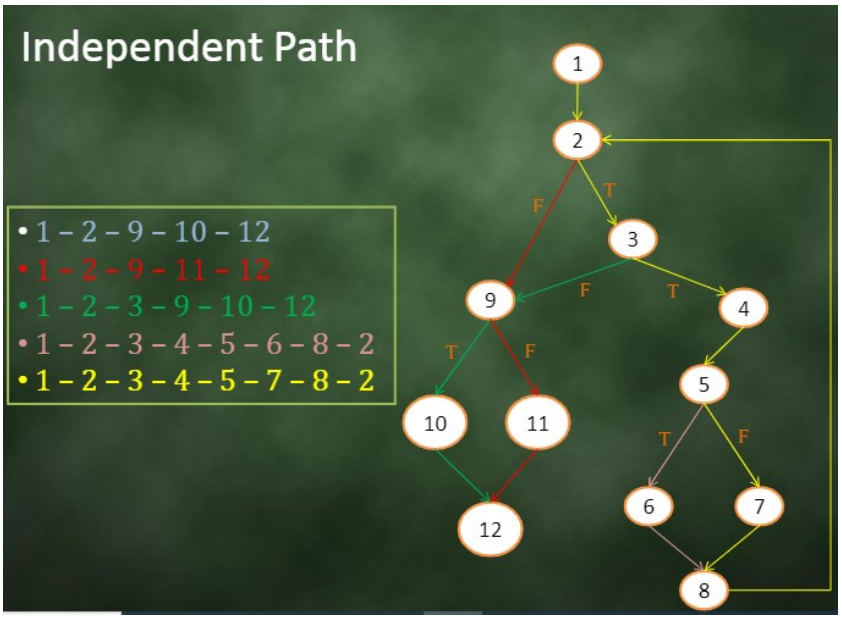
- Mengerjakan seluruh keputusan logika
- Mengerjakan seluruh loop (sesuai batas)
- Menjamin seluruh jalur independen dalam modul dikerjakan minimal 1x
- Mengerjakan seluruh data internal yang menjamin validitas

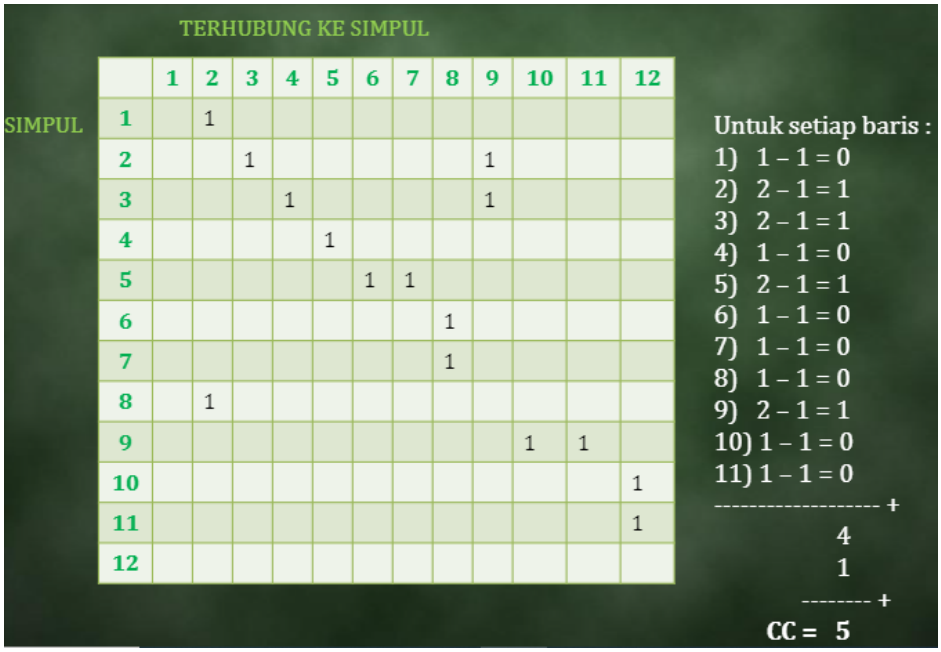
Kekurangan:

- Pengujian secara menyeluruh justru menimbulkan masalah sumber daya
- Program yang kecil bisa menghasilkan banyak sekali jalur logika

Pengujian White-box salah satunya menggunakan teknik **Basis Path Testing** (Berbasis Alur), dimana tujuannya meyakinkan bahwa test case akan menguji setiap path pada suatu program paling sedikit satu kali. Pengujian ini memungkinkan perancangan kasus uji yang diturunkan dari pengukuran tingkat kompleksitas (*cyclomatic complexity*) struktur program. Ukuran tingkat kompleksitas tersebut menjadi panduan di dalam menentukan jalur-jalur dasar (*basis path*). Kasus uji yang didapat menjadi kasus uji yang digunakan untuk membuktikan bahwa setiap pernyataan (*statement*) dari program akan dieksekusi minimal sekali.

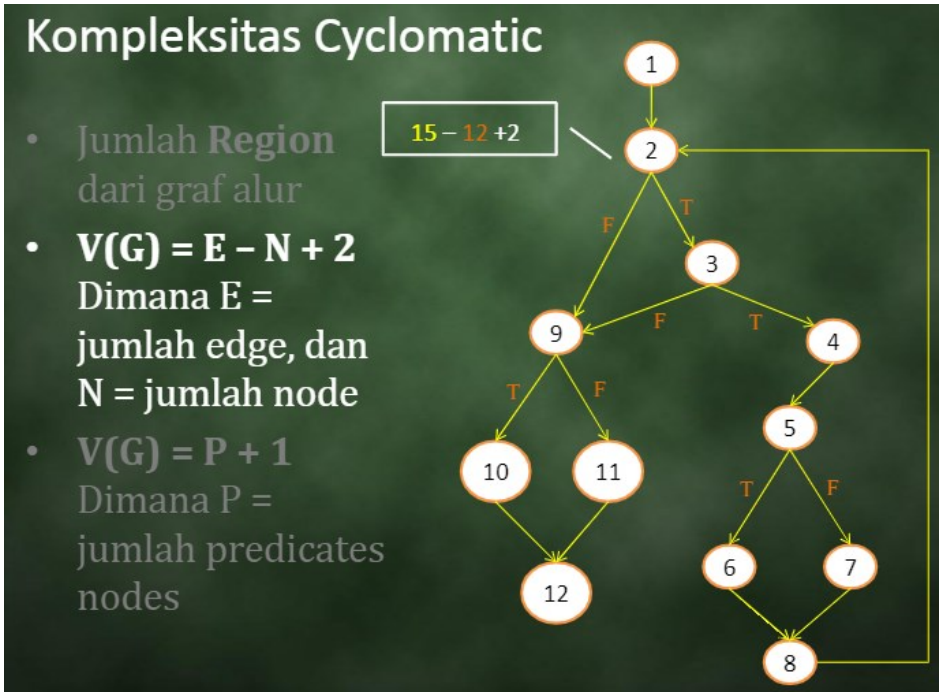






	1	2	3	4	5	6	7	8	9	10	11	12
1		1										
2			1						1			
3				1					1			
4					1							
5						1	1					
6								1				
7								1				
8		1										
9										1	1	
10												1
11												1
12												

Atau menghitung CC (*Cyclomatic Complexity*) dengan rumus V(G) = E-N+2.



V(G) = E-N+2 = 15-12+2 = 5.

4.3 PRAKTIKUM

Contoh

Black-box testing

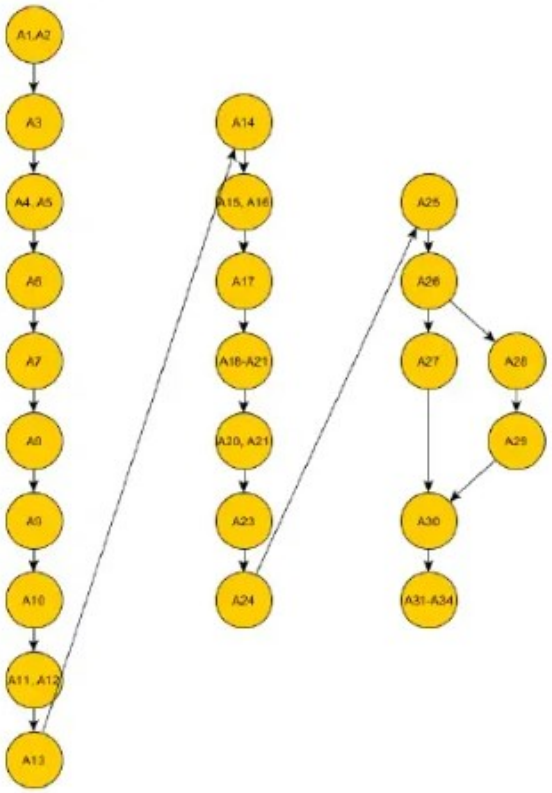
Kasus uji (*test case*) yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login Admin maka kasus uji yang dibuat adalah:

No.	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	*Username* dan *Password* tidak diisi kemudian klik tombol Login	Username: (kosong) Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Harap isi username dan password"	Sesuai harapan	Vaid
2	Mengetikkan Username, dan password tidak diisi atau kosong kemudian klik tombol Login	Username: admin Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Password belum diisi"	Sesuai harapan	Vaid
3	Mengetikkan Password, dan username tidak diisi atau kosong kemudian klik tombol Login	Username: (kosong) Password: admin	Sistem akan menolak dan menampilkan pesan "Username belum diisi"	Sesuai harapan	Vaid
4	Mengetikkan Username dan/atau password tidak sesuai, kemudian klik tombol Login	Username: adm Password: adm123	Sistem akan menolak dan menampilkan pesan "Username atau Password yang anda masukan salah"	Sesuai harapan	Vaid
5	Mengetikkan Username dan password (diisi), kemudian klik tombol Login	Username: admin Password: admin	Sistem menerima akses login dan kemudian menampilkan halaman utama Admin	Sesuai harapan	Vaid

Contoh
White-box testing

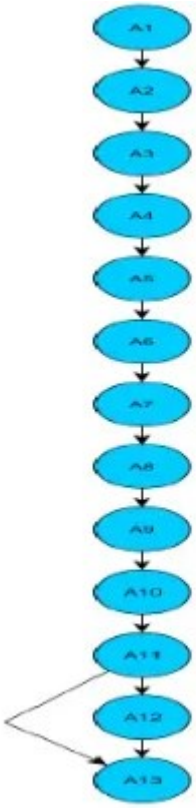
Source code: Index.php

```
1.  <html>
2.  <head>
3.  <link rel="stylesheet" type="text/css" href="style.css">
4.  </head>
5.  <body>
6.  <div class="login">
7.  <form action="login.php" method="post" onSubmit="return validasi()">
8.  <div>
9.  <label>Username:</label>
10. <input type="text" name="username" id="username" />
11. </div>
12.
13. <label>Password:</label>
14. <input type="password" name="password" id="password" />
15. </div>
16. <div>
17. <input type="submit" value="Login" class="tombol">
18. </div>
19. </form>
20. </div>
21. </body>
22. <script type="text/javascript">
23. function validasi() {
24. var username = document.getElementById("username").value;
25. var password = document.getElementById("password").value;
26. if (username != "" && password!="") {
27. return true;
28. }else{
29. alert('Username dan Password harus di isi !');
30. return false;
31. }
32. }
33. </script>
34. </html>
```



Source code: Login.php

```
1.  <?php
2.  include 'config.php';
3.  $username = $_POST['username'];
4.  $password = md5($_POST['password']);
5.  $login = mysql_query("select * from user where
    username='$username' and password='$password'");
6.  $cek = mysql_num_rows($login);
7.  session_start()if($cek > 0){
8.  $_SESSION['username'] = $username;
9.  $_SESSION['status'] = "login";
10. header("location:admin/index.php");
11. }else{
12. header("location:index.php");
13. }?>
```



Selanjutnya analisa hasil halaman pada aplikasi apakah sudah benar dan berfungsi dengan baik.

Praktikum : Lakukan studi literatur melalui beberapa jurnal yang mengimplementasikan masing-masing pendekatan testing tersebut untuk lebih memahami proses pengujian pada kedua metode.

6.3 TUGAS PROGRESS PROJECT AKHIR SEMESTER

1. Diskusikan bersama kelompok untuk studi literatur melalui jurnal, yang mengimplementasikan *Black-Box Testing* (Pengujian kotak hitam) dan *White-Box Testing* (Pengujian kotak putih).

Masing- masing 1 jurnal, dengan ketentuan jurnal berbeda dengan kelompok yang lain. Jika sama nilai akan dibagi.

2. Jurnal tersebut di upload pada G-drive berikut paling lambat 24 Mei 2023. Link Gdrive: https://drive.google.com/drive/folders/1WrabYlcGgJO-6B5N0jXt1kajlzdj0jzZ?usp=share_link

MODUL 9
Pengujian Perangkat Lunak

6.1 Bahasan dan Tujuan

6.1.1 Bahasan

Membahas tahapan pada proses pengujian perangkat lunak dan beberapa macam contoh metode pada pengujian perangkat lunak.

6.1.2 Tujuan

1. Mahasiswa memahami proses pengujian perangkat lunak
2. Mahasiswa mampu merancang kasus uji
3. Mahasiswa mampu memahami dan melakukan pengujian *black box testing* dan *white box testing*

6.2 DASAR TEORI

6.2.1 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya. Kualitas suatu perangkat lunak sangat bergantung pada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan sebagai berikut :

a. Agar dapat "*survive*" bertahan hidup di dunia bisnis perangkat lunak.
b. Dapat bersaing dengan perangkat lunak yang lain.
c. Penting untuk pemasaran global (*global marketing*).
d. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran produksi.
e. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak berada di tangan user. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan "***bug".*** Untuk menghindari banyaknya bug maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan.

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi dan validasi / *verification and validation* (V&V).

a) **Verifikasi :** Sekumpulan aktifitas yang menjamin bahwa software telah diimplentasikan dengan benar untuk suatu fungsi tertentu **(*Are we building the product right ?*)**

b) **Validasi :** Sekumpulan aktifitas yang menjamin bahwa suatu software yang dibuat telah sesuai dengan kebutuhan user **(Are we building the right product?)**

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

1. *Black-Box Testing* (Pengujian kotak hitam)
2. *White-Box Testing* (Pengujian kotak putih)

6.2.2 Black-Box Testing

Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan.

6.2.3 *White-Box Testing*

Pengujian kotak putih dilakukan dengan memeriksa lojik (*logical path*) dari kode program (*source code*) dan desain program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang seharusnya.

Contoh dari *white-box testing* misalkan menguji alur Loop (*Looping Testing*), alur Conditions (*Condition Branch*).

Keuntungan:

Menghasilkan program yang benar dan sempurna 100%,

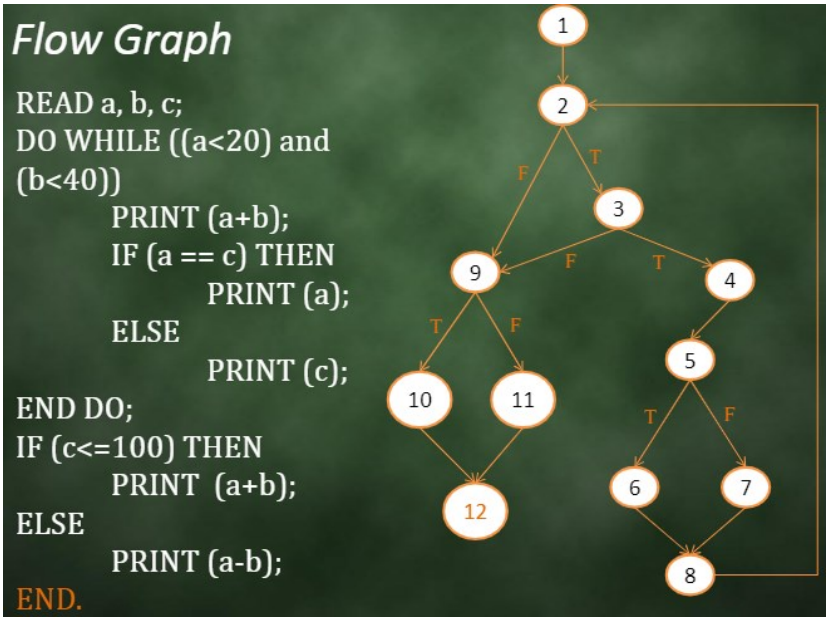
karena:

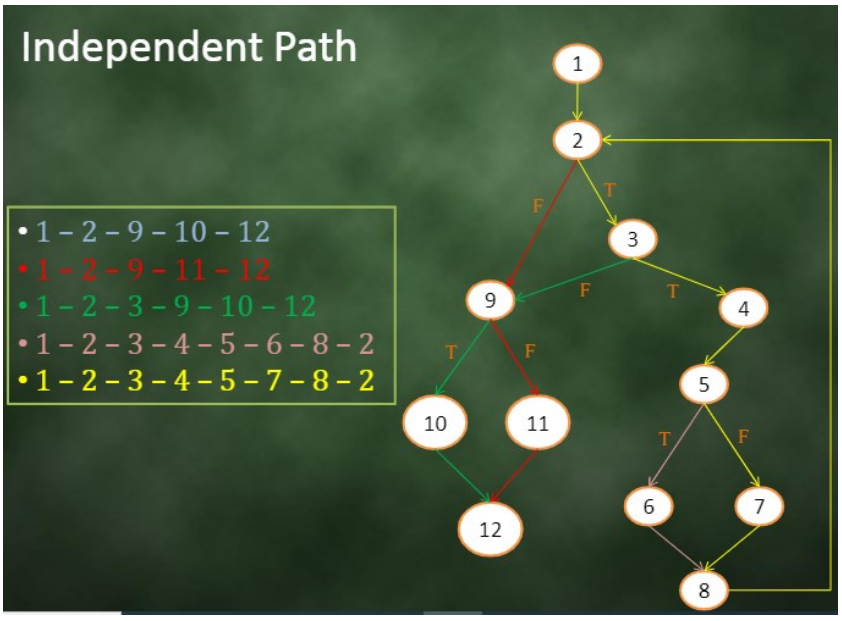
- Mengerjakan seluruh keputusan logika
- Mengerjakan seluruh loop (sesuai batas)
- Menjamin seluruh jalur independen dalam modul dikerjakan minimal 1x
- Mengerjakan seluruh data internal yang menjamin validitas

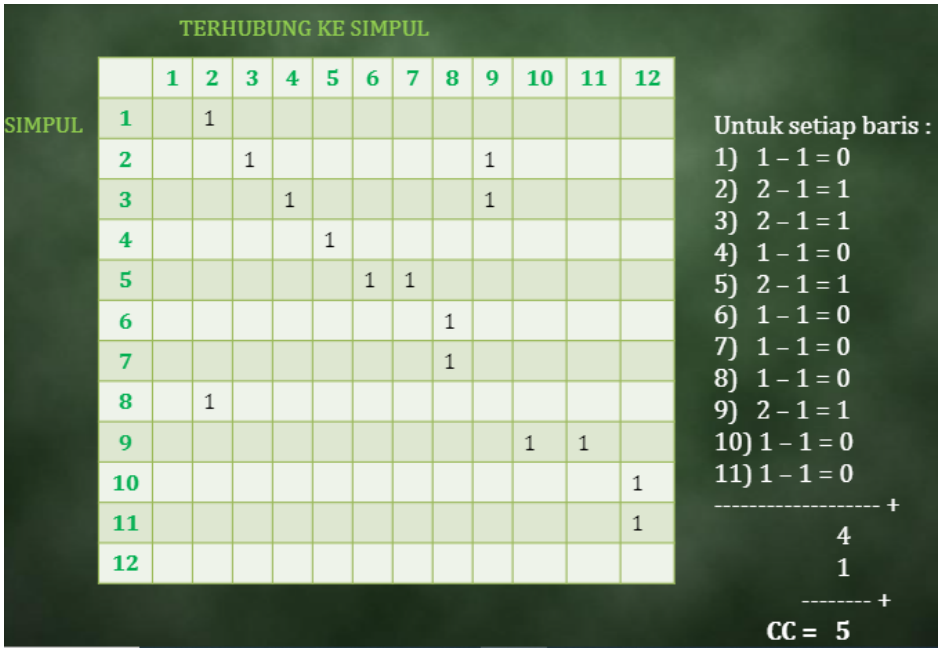
Kekurangan:

- Pengujian secara menyeluruh justru menimbulkan masalah sumber daya
- Program yang kecil bisa menghasilkan banyak sekali jalur logika

Pengujian White-box salah satunya menggunakan teknik **Basis Path Testing** (Berbasis Alur), dimana tujuannya meyakinkan bahwa test case akan menguji setiap path pada suatu program paling sedikit satu kali. Pengujian ini memungkinkan perancangan kasus uji yang diturunkan dari pengukuran tingkat kompleksitas (*cyclomatic complexity*) struktur program. Ukuran tingkat kompleksitas tersebut menjadi panduan di dalam menentukan jalur-jalur dasar (*basis path*). Kasus uji yang didapat menjadi kasus uji yang digunakan untuk membuktikan bahwa setiap pernyataan (*statement*) dari program akan dieksekusi minimal sekali.

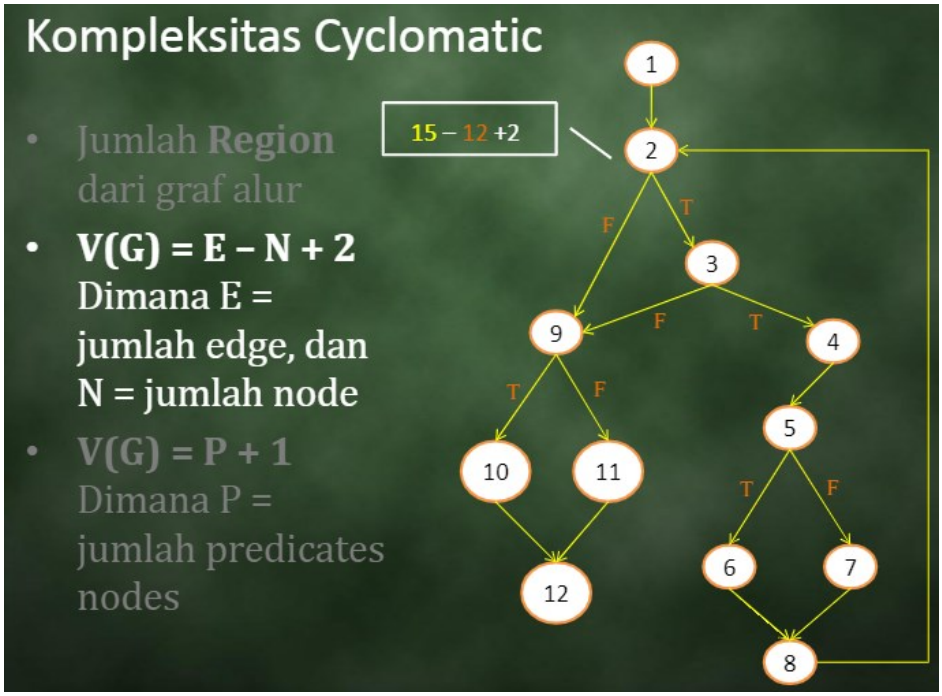






	1	2	3	4	5	6	7	8	9	10	11	12
1		1										
2			1						1			
3				1					1			
4					1							
5						1	1					
6								1				
7								1				
8		1										
9										1	1	
10												1
11												1
12												

Atau menghitung CC (*Cyclomatic Complexity*) dengan rumus V(G) = E-N+2.



V(G) = E-N+2 = 15-12+2 = 5.

4.3 PRAKTIKUM

Contoh

Black-box testing

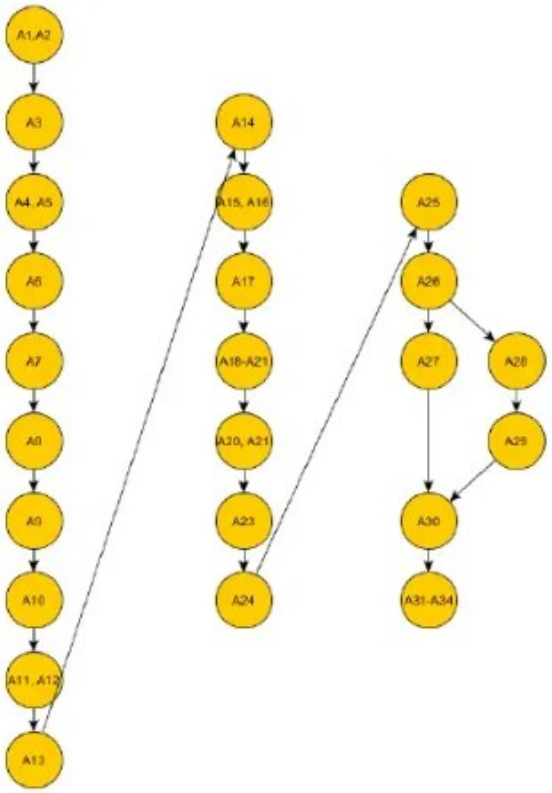
Kasus uji (*test case*) yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login Admin maka kasus uji yang dibuat adalah:

No.	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	*Username* dan *Password* tidak diisi kemudian klik tombol Login	Username: (kosong) Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Harap isi username dan password"	Sesuai harapan	Vaid
2	Mengetikkan Username, dan password tidak diisi atau kosong kemudian klik tombol Login	Username: admin Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Password belum diisi"	Sesuai harapan	Vaid
3	Mengetikkan Password, dan username tidak diisi atau kosong kemudian klik tombol Login	Username: (kosong) Password: admin	Sistem akan menolak dan menampilkan pesan "Username belum diisi"	Sesuai harapan	Vaid
4	Mengetikkan Username dan/atau password tidak sesuai, kemudian klik tombol Login	Username: adm Password: adm123	Sistem akan menolak dan menampilkan pesan "Username atau Password yang anda masukan salah"	Sesuai harapan	Vaid
5	Mengetikkan Username dan password (diisi), kemudian klik tombol Login	Username: admin Password: admin	Sistem menerima akses login dan kemudian menampilkan halaman utama Admin	Sesuai harapan	Vaid

Contoh
White-box testing

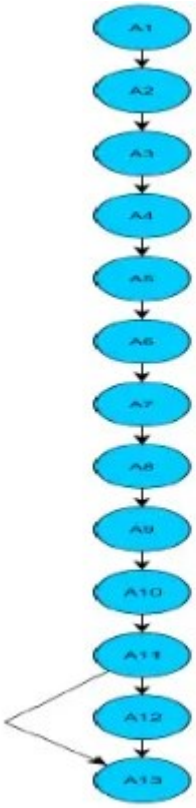
Source code: Index.php

```
1.  <html>
2.  <head>
3.  <link rel="stylesheet" type="text/css" href="style.css">
4.  </head>
5.  <body>
6.  <div class="login">
7.  <form action="login.php" method="post" onSubmit="return validasi()">
8.  <div>
9.  <label>Username:</label>
10. <input type="text" name="username" id="username" />
11. </div>
12.
13. <label>Password:</label>
14. <input type="password" name="password" id="password" />
15. </div>
16. <div>
17. <input type="submit" value="Login" class="tombol">
18. </div>
19. </form>
20. </div>
21. </body>
22. <script type="text/javascript">
23. function validasi() {
24. var username = document.getElementById("username").value;
25. var password = document.getElementById("password").value;
26. if (username != "" && password!="") {
27. return true;
28. }else{
29. alert('Username dan Password harus di isi !');
30. return false;
31. }
32. }
33. </script>
34. </html>
```



Source code: Login.php

```
1.  <?php
2.  include 'config.php';
3.  $username = $_POST['username'];
4.  $password = md5($_POST['password']);
5.  $login = mysql_query("select * from user where
    username='$username' and password='$password'");
6.  $cek = mysql_num_rows($login);
7.  session_start()if($cek > 0){
8.  $_SESSION['username'] = $username;
9.  $_SESSION['status'] = "login";
10. header("location:admin/index.php");
11. }else{
12. header("location:index.php");
13. }?>
```



Selanjutnya analisa hasil halaman pada aplikasi apakah sudah benar dan berfungsi dengan baik.

Praktikum : Lakukan studi literatur melalui beberapa jurnal yang mengimplementasikan masing-masing pendekatan testing tersebut untuk lebih memahami proses pengujian pada kedua metode.

6.3 TUGAS PROGRESS PROJECT AKHIR SEMESTER

1. Diskusikan bersama kelompok untuk studi literatur melalui jurnal, yang mengimplementasikan *Black-Box Testing* (Pengujian kotak hitam) dan *White-Box Testing* (Pengujian kotak putih).

Masing- masing 1 jurnal, dengan ketentuan jurnal berbeda dengan kelompok yang lain. Jika sama nilai akan dibagi.

2. Jurnal tersebut di upload pada G-drive berikut paling lambat 24 Mei 2023. Link Gdrive: https://drive.google.com/drive/folders/1xI08EoQrWgXaJ81d_X4a9VRXSAPDGY9B?usp=share_link

MODUL 9
Pengujian Perangkat Lunak

6.1 Bahasan dan Tujuan

6.1.1 Bahasan

Membahas tahapan pada proses pengujian perangkat lunak dan beberapa macam contoh metode pada pengujian perangkat lunak.

6.1.2 Tujuan

1. Mahasiswa memahami proses pengujian perangkat lunak
2. Mahasiswa mampu merancang kasus uji
3. Mahasiswa mampu memahami dan melakukan pengujian *black box testing* dan *white box testing*

6.2 DASAR TEORI

6.2.1 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya. Kualitas suatu perangkat lunak sangat bergantung pada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan sebagai berikut :

a. Agar dapat "*survive*" bertahan hidup di dunia bisnis perangkat lunak.
b. Dapat bersaing dengan perangkat lunak yang lain.
c. Penting untuk pemasaran global (*global marketing*).
d. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran produksi.
e. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak berada di tangan user. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan "***bug".*** Untuk menghindari banyaknya bug maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan.

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi dan validasi / *verification and validation* (V&V).

a) **Verifikasi :** Sekumpulan aktifitas yang menjamin bahwa software telah diimplentasikan dengan benar untuk suatu fungsi tertentu **(*Are we building the product right ?*)**

b) **Validasi :** Sekumpulan aktifitas yang menjamin bahwa suatu software yang dibuat telah sesuai dengan kebutuhan user **(Are we building the right product?)**

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

1. *Black-Box Testing* (Pengujian kotak hitam)
2. *White-Box Testing* (Pengujian kotak putih)

6.2.2 Black-Box Testing

Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan.

6.2.3 *White-Box Testing*

Pengujian kotak putih dilakukan dengan memeriksa lojik (*logical path*) dari kode program (*source code*) dan desain program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang seharusnya.

Contoh dari *white-box testing* misalkan menguji alur Loop (*Looping Testing*), alur Conditions (*Condition Branch*).

Keuntungan:

Menghasilkan program yang benar dan sempurna 100%,

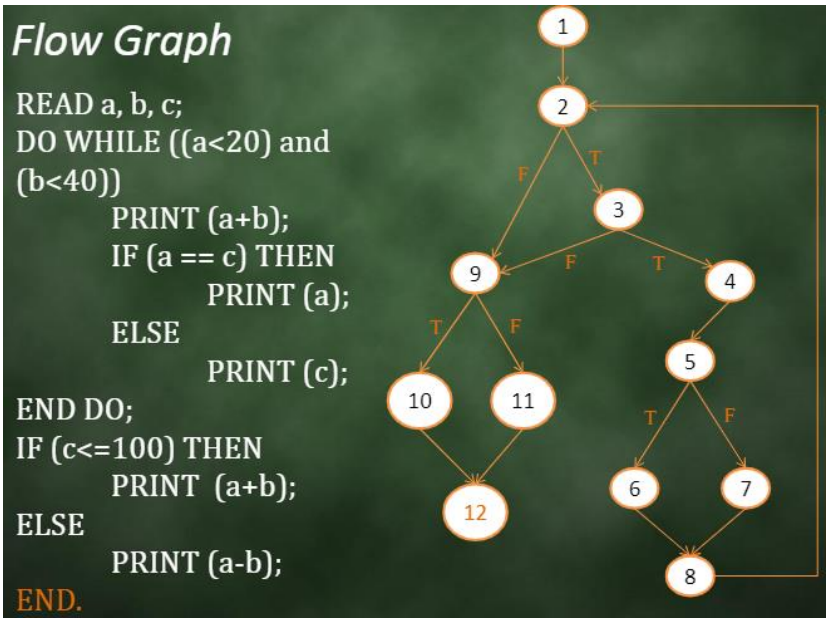
karena:

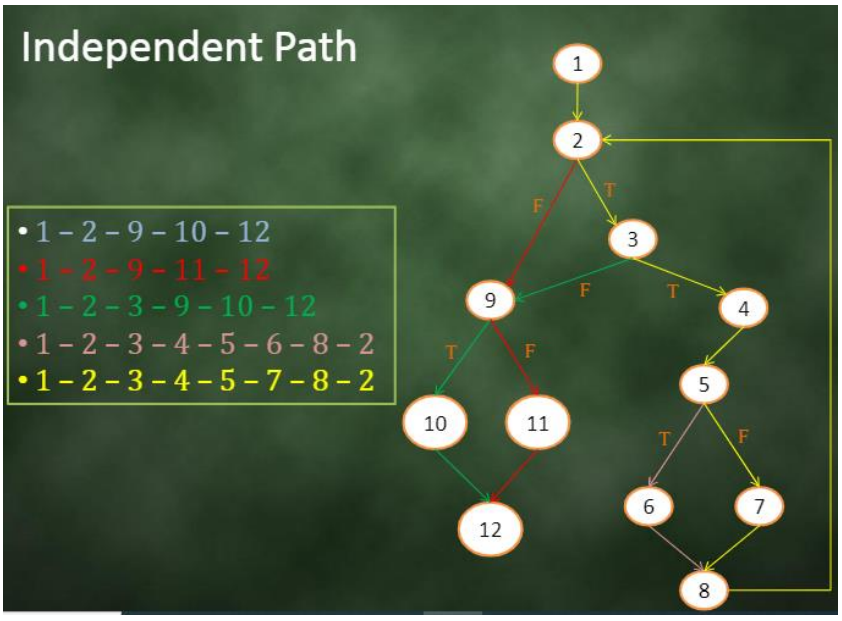
- Mengerjakan seluruh keputusan logika
- Mengerjakan seluruh loop (sesuai batas)
- Menjamin seluruh jalur independen dalam modul dikerjakan minimal 1x
- Mengerjakan seluruh data internal yang menjamin validitas

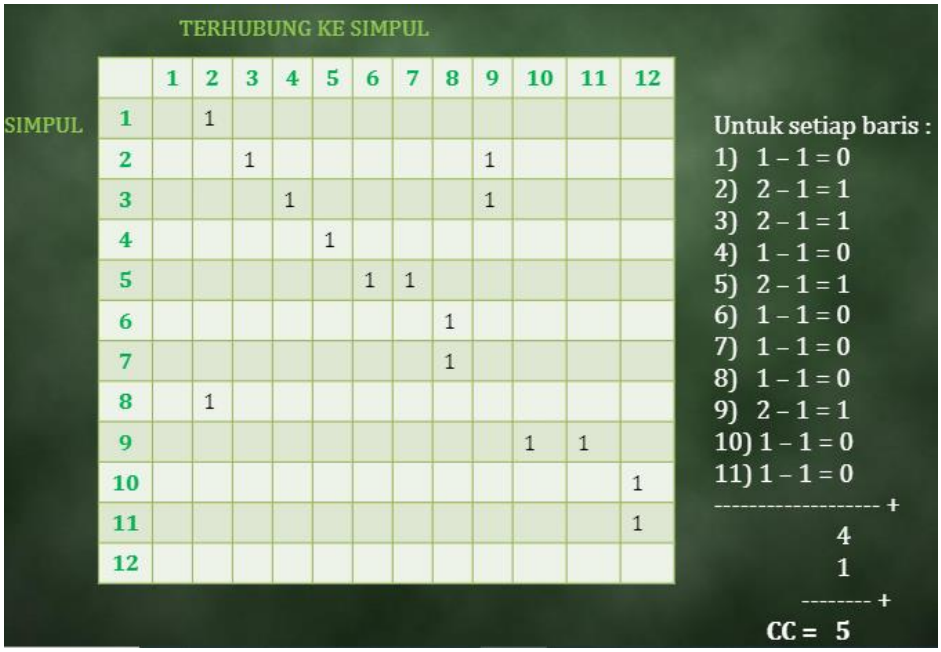
Kekurangan:

- Pengujian secara menyeluruh justru menimbulkan masalah sumber daya
- Program yang kecil bisa menghasilkan banyak sekali jalur logika

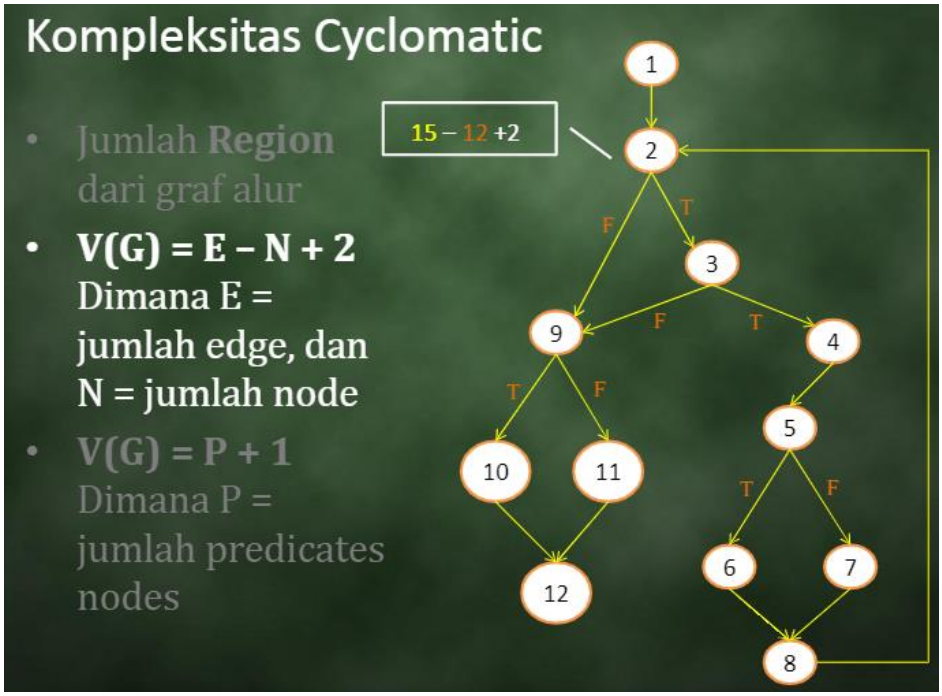
Pengujian White-box salah satunya menggunakan teknik **Basis Path Testing** (Berbasis Alur), dimana tujuannya meyakinkan bahwa test case akan menguji setiap path pada suatu program paling sedikit satu kali. Pengujian ini memungkinkan perancangan kasus uji yang diturunkan dari pengukuran tingkat kompleksitas (*cyclomatic complexity*) struktur program. Ukuran tingkat kompleksitas tersebut menjadi panduan di dalam menentukan jalur-jalur dasar (*basis path*). Kasus uji yang didapat menjadi kasus uji yang digunakan untuk membuktikan bahwa setiap pernyataan (*statement*) dari program akan dieksekusi minimal sekali.







Atau menghitung CC (*Cyclomatic Complexity*) dengan rumus V(G) = E-N+2.



V(G) = E-N+2 = 15-12+2 = 5.

4.3 PRAKTIKUM

Contoh

Black-box testing

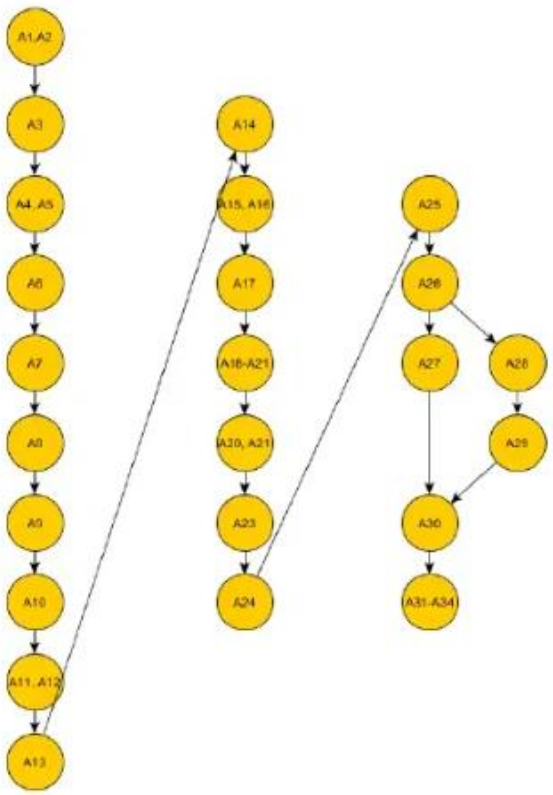
Kasus uji (*test case*) yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login Admin maka kasus uji yang dibuat adalah:

No.	Skenario Pengujian	*Test Case*	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	*Username* dan *Password* tidak diisi kemudian klik tombol Login	Username: (kosong) Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Harap isi username dan password"	Sesuai harapan	Vaid
2	Mengetikkan Username, dan password tidak diisi atau kosong kemudian klik tombol Login	Username: admin Password: (kosong)	Sistem akan menolak dan menampilkan pesan "Password belum diisi"	Sesuai harapan	Vaid
3	Mengetikkan Password, dan username tidak diisi atau kosong kemudian klik tombol Login	Username: (kosong) Password: admin	Sistem akan menolak dan menampilkan pesan "Username belum diisi"	Sesuai harapan	Vaid
4	Mengetikkan Username dan/atau password tidak sesuai, kemudian klik tombol Login	Username: adm Password: adm123	Sistem akan menolak dan menampilkan pesan "Username atau Password yang anda masukan salah"	Sesuai harapan	Vaid
5	Mengetikkan Username dan password (diisi), kemudian klik tombol Login	Username: admin Password: admin	Sistem menerima akses login dan kemudian menampilkan halaman utama Admin	Sesuai harapan	Vaid

Contoh
White-box testing

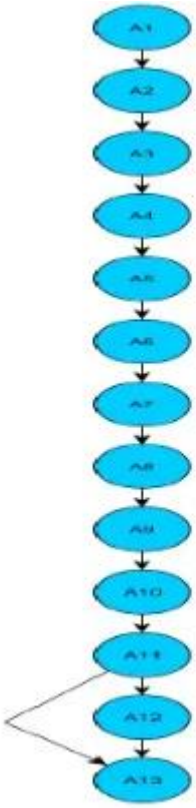
Source code: Index.php

```
1.  <html>
2.  <head>
3.  <link rel="stylesheet" type="text/css" href="style.css">
4.  </head>
5.  <body>
6.  <div class="login">
7.  <form action="login.php" method="post" onSubmit="return validasi()">
8.  <div>
9.  <label>Username:</label>
10. <input type="text" name="username" id="username" />
11. </div>
12.
13. <label>Password:</label>
14. <input type="password" name="password" id="password" />
15. </div>
16. <div>
17. <input type="submit" value="Login" class="tombol">
18. </div>
19. </form>
20. </div>
21. </body>
22. <script type="text/javascript">
23. function validasi() {
24. var username = document.getElementById("username").value;
25. var password = document.getElementById("password").value;
26. if (username != "" && password!="") {
27. return true;
28. }else{
29. alert('Username dan Password harus di isi !');
30. return false;
31. }
32. }
33. </script>
34. </html>
```



Source code: Login.php

```
1.  <?php
2.  include 'config.php';
3.  $username = $_POST['username'];
4.  $password = md5($_POST['password']);
5.  $login = mysql_query("select * from user where
    username='$username' and password='$password'");
6.  $cek = mysql_num_rows($login);
7.  session_start()if($cek > 0){
8.  $_SESSION['username'] = $username;
9.  $_SESSION['status'] = "login";
10. header("location:admin/index.php");
11. }else{
12. header("location:index.php");
13. }?>
```



Selanjutnya analisa hasil halaman pada aplikasi apakah sudah benar dan berfungsi dengan baik.

Praktikum : Buatlah pengujian PL dengan metode *black-box testing* dan *white-box testing* atau kombinasi keduanya dari perangkat lunak yang Anda kembangkan (Tugas Individu) untuk lebih memahami perbedaan kedua pendekatan pengujian tersebut.

Catatan :

- Fungsi perangkat lunak secara garis besar saja (fungsi utama).
- Lakukan studi literatur melalui beberapa jurnal yang mengimplementasikan masing-masing pendekatan testing tersebut untuk lebih memahami proses pengujian pada kedua metode.

6.4 TUGAS PROGRESS PROJECT AKHIR SEMESTER

1. Diskusikan bersama kelompok untuk memutuskan pendekatan pengujian PL yang tepat dari aplikasi yang kelompok Anda kembangkan! (lanjutan tugas sebelumnya).
2. Buatlah rincian pengujian pada perangkat lunak tersebut!

MODUL 10
Business Process Modeling (BPM)

10.1 Bahasan dan Tujuan

10.1.1 Bahasan

Membahas tentang *business process modeling* beserta notasi yang ada di dalamnya.

10.1.2 Tujuan

1. Mahasiswa memahami *business process modeling* pada proses bisnis.
2. Mahasiswa mampu merancang proses bisnis menggunakan *Business Process Modeling Notation* (BPMN).

10.2 Dasar Teori

10.2.1 *Business Process Modeling*

Sebelum memulai sebuah bisnis baik dalam skala kecil maupun besar, diperlukan rancangan atau arsitektur dari sebuah proses bisnis yang akan dijalankan. Proses bisnis harus memiliki sumber daya dan melibatkan berbagai pihak yang berkepentingan. Pengelolaan bisnis yang tepat dapat meningkatkan kinerja suatu organisasi secara keseluruhan sehingga tingkat kepuasan konsumen dapat tercapai dengan baik.

Workflow Management Coalition (WfMC) mendefinisikan proses bisnis sebagai kumpulan prosedur atau aktivitas yang dapat mendefinisikan objektif atau tujuan bisnis, umumnya dalam konteks struktur organisasi yang dapat mendefinisikan peranan dan hubungan fungsional pada organisasi tersebut. Dokumentasi akan proses bisnis salah satunya dapat dilakukan dengan membuat diagram *Business Process Modeling* menggunakan *Business Process Modeling Notation*.

10.2.2 *Business Process Modeling Notation*

Business Process Modeling Notation atau yang disingkat dengan BPMN, yaitu sebuah standar untuk menggambarkan proses bisnis yang dikeluarkan oleh *Open Management Group* (omg.org). BPMN sebagai suatu standar baru pada pemodelan proses bisnis, dan juga sebagai alat desain pada sistem yang kompleks seperti sistem *e-Business* yang berbasis pesan (*message-based*). Secara sederhana, BPMN adalah notasi grafis yang menggambarkan logika dari langkah-langkah dalam proses bisnis. Notasi ini telah didesain secara khusus untuk mengkoordinasikan urutan proses dan pesan yang mengalir antara pelaku dalam kegiatan yang berbeda.

Tujuan utama dari BPMN adalah menyediakan notasi yang mudah digunakan dan dimengerti oleh semua orang yang terlibat dalam bisnis, yang meliputi bisnis analis yang memodelkan proses bisnis, pengembang teknik yang membangun sistem yang melaksanakan bisnis, dan berbagai tingkatan manajemen yang harus dapat membaca dan memahami proses diagram dengan cepat sehingga dapat membantu dalam pengambilan keputusan. BPMN juga menciptakan suatu jembatan terstandardisasi untuk *gap* antara desain proses bisnis dan implementasi proses.

1. Pentingnya BPMN

- BPMN adalah standar proses pemodelan diterima secara internasional.
- BPMN adalah suatu metodologi pemodelan proses.
- BPMN menciptakan jembatan standar yang mengurangi kesenjangan antara proses bisnis dan pelaksanaannya.
- BPMN memungkinkan setiap orang dalam organisasi dapat saling memahami proses bisnis.

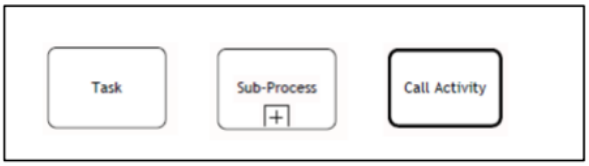
2. Notasi BPMN

Notasi BPMN dibagi menjadi lima kategori dasar. Lima kategori dasar elemen BPMN yaitu ***Flow Objects***, ***Data***, ***Connecting Objects***, ***Swimlanes***, ***Artifacts***.

a. ***Flow Objects***, elemen grafis utama untuk menentukan perilaku dalam Proses Bisnis. Terdapat tiga *Flow Object* yaitu ***events, activities, gateways***.

- ***Events***, sesuatu yang "terjadi" selama jalannya Proses. Mempengaruhi aliran dari model dan bisanya memiliki penyebab (pemicu) atau dampak (hasil). *Event* digambarkan dalam lingkaran terbuka untuk membedakan fungsinya. Terdapat tiga jenis *event*, berdasarkan pengaruh aliran proses yaitu Awal, Menengah, dan Akhir.

- ***Activities***, suatu kegiatan yang memperlihatkan organisasi melakukan Proses. *Activities* digambarkan dengan bentuk persegi panjang.

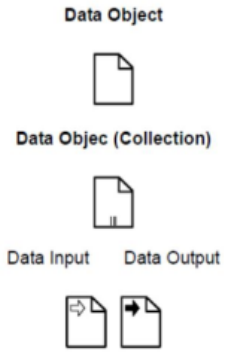


- ***Gateways***, mendefinisikan semua tindakan arus urutan proses bisnis.

Exclusive (XOR), Parallel (AND), Inclusive (OR) Gateway

b. ***Data***, data direpresentasikan dengan empat elemen yaitu ***Data Objects***, ***Data Inputs***, ***Data Inputs***, dan ***Data Stores***.

Data Objects, memberikan informasi tentang kegiatan apa yang perlu diadakandan atau apa yang mereka hasilkan. *Data Object* dapat mewaakili benda tunggal atau koleksi benda-benda. Data input dan Data Output memberikan informasi yang sama untuk Proses.



c. ***Connecting Objects***, *connecting object* digunakan untuk menghubungkan obyek arus informasi satu sama lain. Terdapat empat model *connecting object* yaitu ***Sequence Flows***, ***Message Flows***, ***Associations***, dan ***Data Associations***.

- ***Sequence Flows***, sebuah arus urutan digunakan untuk menunjukkan urutan kegiatan yang akan dilakukan dalam proses.

- ***Message Flows***, digunakan untuk menunjukan aliran pesan antara dua pelaku yang telah dipersiapkan untuk mengirim dan menerima.

- ***Associations***, digunakan untuk menghubungkan informasi dan Artefak dengan elemen BPMN grafis.

d. ***Swimlines***, terdapat dua jenis pemodelan utama menggunakan *swimlines* yaitu ***pools*** dan ***lanes***.

- ***Pool*,** representasi grafis dari pelaku/peserta kolaborasi.

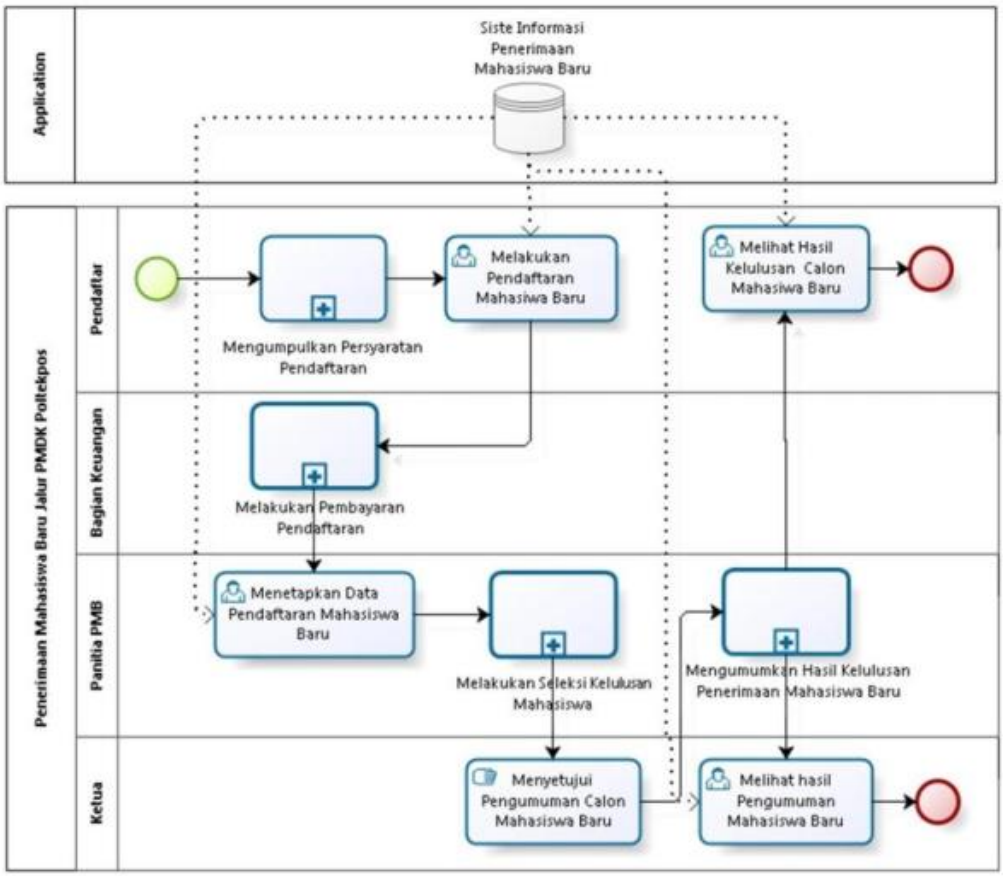
- ***Lanes***, partisi sub-dalam proses, terkadang dalam Pool, akan memperpanjang seluruh proses baik secara vertikal ataupun horisontal. Jalur yang digunakan untuk mengatur dan mengkategorikan kegiatan.

e. ***Artifacts***, digunakan untuk memberikan informasi tambahan tentang Proses. Terdapat dua *artifacts* standar yaitu ***group*** dan ***text annotation***.

10.3 Kegiatan Praktikum

Contoh 1:

Berikut adalah contoh BPMN untuk proses bisnis pendaftaran mahasiswa.



Keterangan untuk gambar di atas :
1. Simbol Database yang disebut dengan artifak yang menunjukan bahwa data yang ada masuk kedalam sistem.
2. Simbol bulat berwarna hijau menandakan awal proses dan merah sebagai akhir proses.
3. Tanda (+)plus pada kotak proses menunjukan didalam proses tersebut masih ada proses lain di dalamnya.

Contoh 2:
Contoh BPMN berikutnya yaitu BPMN untuk bisnis proses layanan pizza. Penggambaran BPMN antara pelanggan pizza dan vendor pizza menunjukkan alur proses yang terjadi pada pemesanan pizza hingga pembayaran pesanan pizza oleh pelanggan pizza kepada vendor.