BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Tempat Penelitian

Tinjauan tempat penelitian bertujuan untuk mengetahui keadaan yang ada di tempat penelitian di antaranya adalah sejarah berdirinya, visi dan misi, dan struktur organisasi dari pihak yang nantinya terlibat di dalam sistem.

2.1.1 Sejarah Singkat Universitas Komputer Indonesia

Rektor Universitas Komputer Indonesia, Dr. Ir. Eddy Suryanto Soegoto, mendirikan Lembaga Pendidikan Komputer Indonesia Jerman (LPKIG) pada tahun 1994 untuk program pendidikan 1 tahun, lembaga tersebut berlokasi di Jalan Dipati Ukur No. 102. Pada tahun ketiga, nama Lembaga Pendidikan Komputer Indonesia Jerman (LPKIG) diubah menjadi Indonesian *Germany Institute* (IGI). Pada tanggal 24 Desember 1998 dibentuk Yayasan *Science* dan Teknologi yang dilanjutkan dengan Pengajuan ke Kopertis IV Jabar untuk pendirian STIE IGI dan STMIK IGI. Pada bulan Agustus 1999 keluar SK Mendiknas No. 143/D/O/1999 atas STIMK IGI dengan 5 Program Studi yaitu Teknik Informatika S1, Manajemen Informatika D3 & D1, Teknik Komputer D3 & D1, Teknik Informatika D3 & D1, Komputerisasi Akuntansi D3 & D1[8].

Mengantisipasi pesatnya perkembangan Teknologi Informasi, IPTEK, Era Globalisasi dan Milenium ke-3 serta untuk memberikan yang terbaik bagi masa depan mahasiswa/i, Yayasan *Science* dan Teknologi kemudian mengajukan usulan ke DIKTI melalui Kopertis Wilayah IV Jabar untuk melakukan merger atas kedua Sekolah Tinggi di atas untuk menjadi "Universitas Komputer Indonesia". Pada hari Selasa, tanggal 8 Agustus 2000 keluar SK Mendiknas No. 126/D/O/2000 atas nama Universitas Komputer Indonesia, disingkat UNIKOM. Akreditasi Mendiknas atas UNIKOM disertai menetapkan kembali status Terdaftar kepada Program Studi Teknik Informatika jenjang pendidikan program S1 (Sarjana)[8].

2.1.2 Visi, Misi, Tujuan, Budaya Organisasi, dan Motto

- **Visi**

  Menjadi Universitas terkemuka dibidang Teknologi Informasi & Komunikasi, berwawasan Global, berjiwa *Entrepreneur* dan menjadi Pusat Unggulan dibidang Ilmu Pengetahuan dan Teknologi yang mendukung Pembangunan Nasional serta berorientasi pada kepentingan Masyarakat, Bangsa dan Negara[8].

- **Misi**

  Menyelenggarakan Pendidikan Tinggi Modern berdasarkan Budaya Organisasi UNIKOM, PIQIE (*Professionalism, Integrity, Quality, Information Technology, Excellence*), dengan Sistem Pendidikan yang Kondusif dan Program–program Studi yang berbasis pada *Software* (Perangkat Lunak), Hardware (Perangkat Keras), dan *Entrepreneurship* (Kewirausahaan) dengan mengoptimalkan Sumber Daya yang ada berdasarkan prinsip Efisiensi, Efektivitas dan Produktivitas[8].

- **Tujuan**

  Menghasilkan Lulusan yang unggul dibidang Teknologi Informasi & Komunikasi, Kompeten dan Handal di Bidang Studinya, berjiwa *Entrepreneur*, Santun dan Berbudi Luhur, Memiliki Komitmen untuk memajukan Bangsa dan Negara serta Beriman dan Bertaqwa kepada Tuhan Yang Maha Esa[8].

- **Budaya Organisasi**

  PIQIE (*Professionalism, Integrity, Quality, Information Technology, Excellence*)[8].

- **Motto**

  Quality Is Our Tradition[8].

2.1.3 Logo Universitas Komputer Indonesia

Logo dari suatu universitas akan menggambarkan citra dari universitas itu sendiri. Logo UNIKOM dapat dilihat pada Gambar 2.1-1.

Gambar 2.1-1 Logo Universitas Komputer Indonesia

Adapun makna dari simbol/gambar pada logo UNIKOM sebagai berikut :

- **Bingkai Segi Lima**

  Melambangkan UNIKOM berlandaskan falsafah negara yakni Pancasila dan Undang-Undang Dasar 1945[8].

- **Lingkaran Dalam Segi Lima Tempat Tulisan Berwarna Kuning**

  Melambangkan *motto* UNIKOM menuju kejayaan yakni *Quality Is Our Tradition*[8].

- **Bulatan Dalam Berwarna Biru**

  Melambangkan UNIKOM bertujuan menghasilkan ilmuwan unggul dan berpikiran maju yang Bertaqwa kepada Tuhan Yang Maha Esa[8].

- **Komputer**

  Melambangkan ciri utama UNIKOM yang memberikan pendidikan Teknologi Informasi dan Komputasi pada seluruh Jurusan yang ada dilingkungan Universitas Komputer Indonesia, menjadi Universitas Terdepan dibidang Teknologi Informasi dan Komputer serta sebagai Universitas komputer pertama di Indonesia[8].

- **Stasiun *Relay***

  Melambangkan UNIKOM menyelenggarakan Pendidikan Tinggi ke arah masyarakat industri maju dengan sistem pendidikan yang kondusif dan tenaga pengajar berkualitas untuk menghasilkan lulusan-lulusan terbaik[8].

- **Satelit**

  Melambangkan UNIKOM berwawasan Global dan menjadi pusat unggulan dibidang IPTEK & seni yang mendukung Pembangunan Nasional serta berorientasi pada kepentingan masyarakat, bangsa dan negara[8].
- **Cakrawala**

  Melambangkan indahnya menggapai Cita-cita dan mengejar ilmu setinggi Langit[8].
- **Buku**

  Melambangkan sumber ilmu yang tiada habis-habisnya[8].

2.2 Landasan Teori

Subbab landasan teori berisikan teori-teori pendukung yang digunakan pada proses analisis dan implementasi pada penelitian ini untuk pengembangan perangkat lunak di sistem *monitoring loging* di *proxy* UNIKOM. Adapun teori-teori yang digunakan sebagai berikut.

2.2.1 *Reengineering*

Reengineering adalah proses menganalisis sistem yang ada ke dalam bentuk baru untuk meningkatkan kualitas beroperasi, kemampuan sistem, fungsional, kinerja, dan kemampuan untuk berkembang dengan biaya yang murah. Dalam tahap siklus, *reengineering* mencangkup jangkauan luas, mulai dari *existing implementation*, *recapturing* atau *recreation* dari desain, dan mengartikan *requirements* secara nyata dalam implementasi oleh subyek sistem[9].

Reengineering dapat digunakan untuk mengekstraksi informasi-informasi yang ada dalam perangkat lunak. Informasi yang diekstraksi dapat digunakan untuk membangun kembali perangkat lunak ataupun membuat dokumentasi yang lebih ter*update* dan akurat terhadap perangkat lunak.

Saat melakukan *reengineering* ada beberapa faktor yang menjadi penyebab sistem untuk di*reengineering*. Faktor-faktor yang menyebabkan *reengineering* yaitu sebagai berikut[10].

1. Hilangnya atau tidak lengkapnya desain/spesifikasi.
2. Kadaluwarsa, tidak sesuai atau hilangnya dokumentasi.
3. Meningkatnya kompleksitas program
4. Kurang terstrukturnya *source code*.
5. Kebutuhan untuk menerjemahkan program ke dalam bahasa program yang berbeda.

2.2.1.1 Melihat Tingkat Abstraksi *Reengineering*

Tingkat abstraksi *reengineering* merupakan kelengkapan proses yang mengacu pada detail yang diberikan abstraksi(proses representasi data dan program dalam bentuk yang sama dalam implementasi)[11]. Tingkat abstraksi proses *reengineering* dapat diketahui dengan cara melakukan beberapa tahapan yaitu sebagai berikut.

1. Representasikan Prosedural

   Gambaran dari prosedural sistem yang sedang berjalan.
2. Program dan Informasi Struktur Data

   Mengumpulkan informasi kebutuhan sistem dengan cara melakukan wawancara dengan pengguna sistem.
3. Data dan Model *System Control*

   Data yang sudah dikumpulkan dalam wawancara akan dilakukan analisis untuk membuat pemodelan sistem yang sesuai.
4. Model Hubungan Entitas

   Melakukan desain ulang sistem yang sudah di analisis dengan mempelajari peta proses untuk memudahkan dalam melihat bagian-bagian tertentu yang bisa diotomatisasikan dan disederhanakan dengan cara membuat UML.

2.2.1.2 Proses Reengineering

Dalam melakukan *reengineering* pada perangkat lunak memiliki tiga tahapan atau proses utama yang perlu dilakukan[12]. Proses *reengineering* meliputi *reverse engineering, restructuring*, dan *forward engineering.*

a. *Reverse Engineering*

   Reverse engineering adalah proses menganalisis sistem subjek untuk mengidentifikasi komponen sistem dan keterkaitannya serta membuat representasi sistem dalam bentuk lain atau pada tingkat abstraksi yang lebih tinggi. Tahapan dalam *reverse engineering* beberapa tahapan sebagai berikut.

   1. *Implementation (code*)
   2. *Design*
   3. *Requiremnt*

b. *Forward Engineering*

   Forward Engineering adalah sebuah proses pengubahan dari abstraksi level yang paling tinggi(*Requirement*) dan *logic* ke level *design* sampai ke level fisik (Code)dari sistem.

   1. *Requirement*
   2. *Design*
   3. *Implementation (code)*

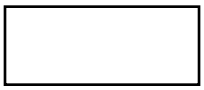
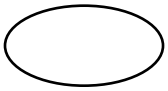
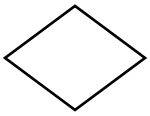
2.2.2 Sistem Basis Data

Sistem basis data secara umum merupakan koleksi dari data-data yang terorganisasi sedemikian rupa sehingga data mudah untuk disimpan dan dimanipulasi (diperbarui, dicari, diolah dengan perhitungan-perhitungan tertentu, serta dihapus). Basis data digunakan untuk menyimpan objek-objek seperti dokumen, citra fotografi, suara, serta video[13]. Hal yang mendasari struktur basis data adalah model data. Model data adalah sekumpulan cara, *tool* untuk mendeskripsikan data-data, hubungannya satu sama lain, serta batasan konsistensi. Model data dibagi menjadi 3 bagian.

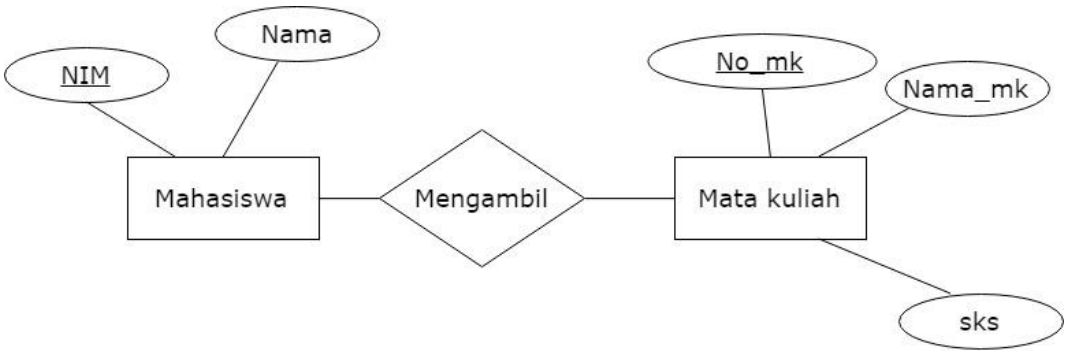
a. Model Data *Conceptual*

Model data *conceptual* merupakan modal data yang menggambarkan hubungan logika antar data dalam basis data berdasarkan objek data. Model data ini berbentuk *Entity Relationship Model* atau ERD. Bentuk simbol ERD dapat dilihat pada Tabel 2.2-1.

Tabel 2.2-1 Simbol ERD

Simbol	Nama	Keterangan
	Entitas	Suatu objek di dunia nyata yang dapat dibedakan dengan objek lain. Contoh Mahasiswa, Pegawai dan lain-lain
	Atribut	Properti deskriptif yang dimiliki oleh setiap anggota dari himpunan entitas. Contoh : NIM, nama, umur, dll.
	Relasi	Hubungan antara suatu entitas dengan entitas lainnya. Contoh : entitas Mahasiswa memiliki hubungan tertentu dengan entitas Mata kuliah.
	Garis	Hubungan antara entitas dengan atributnya dan entitas dengan himpunan relasinya
NIM	Key	Atribut yang digunakan untuk membedakan suatu entitas dengan entitas lainnya dalam himpunan entitas.

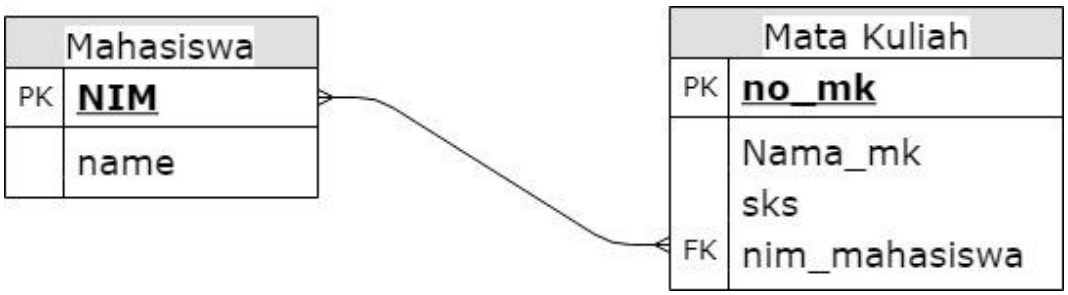
Penggambaran model ERD untuk pengambilan mata kuliah dapat dilihat pada Gambar 2.2-1.



Gambar 2.2-1 Model ERD

b. Model Data *Logical*

Model data *logical* merupakan model data yang menggambarkan hubungan logika antar data dalam basis data berdasarkan *record-record* pada tabel *database*. model data ini berbentuk skema relasi dapat dilihat pada Tabel 2.2-2.



Gambar 2.2-2 Model Skema Relasi

c. Model Data *Physical*

Model data *physical* menjelaskan bagaimana sistem akan diimplementasikan menggunakan sistem DBMS yang berbentuk tabel-tabel dalam *database* dan bagaimana data akan disimpan. Model data ini berbentuk struktur tabel.

1. Tabel Mahasiswa

Tabel 2.2-2 Struktur Tabel Mahasiswa

Atribut	Tipe	*Size*	*Key*	Keterangan
NIM	integer	11	*Primary Key*	NIM Mahasiswa
name	*varchar*	100	-	Nama Mahasiswa

2. Tabel Mata Kuliah

Tabel 2.2-3 Struktur Tabel Mata Kuliah

Atribut	Tipe	*Size*	*Key*	Keterangan
no_mk	integer	11	*Primary Key*	Nomor Mata kuliah
Nama_mk	*varchar*	100	-	Nama Mata kuliah
Sks	Integer	3	-	Sks Matakuliah

2.2.3 Proxy Server

Proxy server merupakan komponen penengah antar *user*, serta bertindak sebagai server dan *client* yang menerima *request message* dari *user* untuk disampaikan pada *user* lainnya. *Request* yang diterima dapat dilayani sendiri atau

disampaikan (*forward*) pada *proxy* lain atau server lain[1]. *Proxy server* juga dapat digunakan untuk mengamankan jaringan pribadi yang dihubungkan ke sebuah jaringan publik (seperti halnya Internet). *Proxy server* memiliki lebih banyak fungsi daripada *router* yang cuma memiliki *fitur packet filtering*, karena *proxy* server beroperasi pada level yang lebih tinggi dan memiliki kontrol yang lebih menyeluruh terhadap akses jaringan. Proxy server yang berfungsi sebagai sebuah "agen keamanan" untuk sebuah jaringan pribadi, umumnya dikenal sebagai *firewall*[14].

Manfaat *Proxy* Server

1. Meningkatkan Kinerja Jaringan

   Proxy server memiliki kemampuan untuk menyimpan data *request* / permintaan dari aplikasi *client*, karena *request* yang sama dengan *request* sebelumnya hanya akan diambilkan dari simpanan server *proxy*. Apabila *user* sudah pernah membuka situs yang sama, maka *user* tidak perlu dihubungkan langsung dengan situs sumbernya, namun cukup diambil data dari simpanan server *proxy*.

2. Memfilter permintaan

   Proxy server dapat juga digunakan sebagai filter untuk *request* data pada suatu situs. Penggunaan filter pada server *proxy* dapat memberikan hak akses terhadap situs mana yang boleh atau tidak boleh dikunjungi oleh *user*. Selain itu *proxy* server juga digunakan sebagai filter untuk berbagai gangguan internet.

Beberapa keuntungan penggunaan *proxy* server dalam suatu jaring TCP/IP yaitu sebagai berikut :

1. Keamanan jaringan komputer akan lebih terjaga, karena adanya pembatas antara jaringan lokal dengan jaringan luar(internet).
2. Akses pada *website-website* yang sudah pernah di akses sebelumnya akan menjadi lebih cepat.
3. Memiliki fungsi *filtering*, baik itu *filtering* pengguna, *content*, atau waktu akses.

Server selain memiliki banyak keuntungan di dalamnya juga memiliki kekurangan dalam suatu jaringan TCP/IP. Kekurangan *proxy* server yaitu sebagai berikut.

1. Akses terhadap *website* yang belum pernah dibuka sebelumnya akan lebih lambat.
2. Jika *proxy* terlambat melakukan *update cache*, maka *client* akan mendapat konten yang belum *terupdate*.

2.2.4 Visualisasi Data

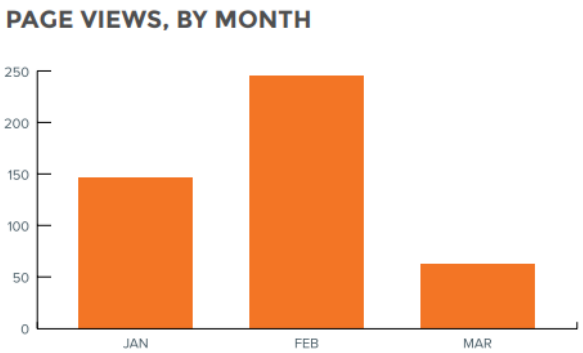
Visualisasi merupakan metode yang sering digunakan untuk memahami data secara efektif dan efisien. Visualisasi data yang efektif akan membantu pengguna dalam melakukan analisis, mempermudah dalam menyimpulkan data dan membuat data yang kompleks menjadi lebih mudah untuk dimengerti[15]. Sebelum memahami visualisasi, jenis data yang dapat divisualisasikan dan hubungan data satu sama lain harus dapat dipahami. Memilih teknik visual yang baik untuk menampilkan kunci data memegang dampak yang baik.

2.2.4.1 Bentuk Visualisasi Data

Bentuk visualisasi dalam menyampaikan informasi kepada pengguna terbagi dalam beberapa bentuk grafik yaitu sebagai berikut[16][17].

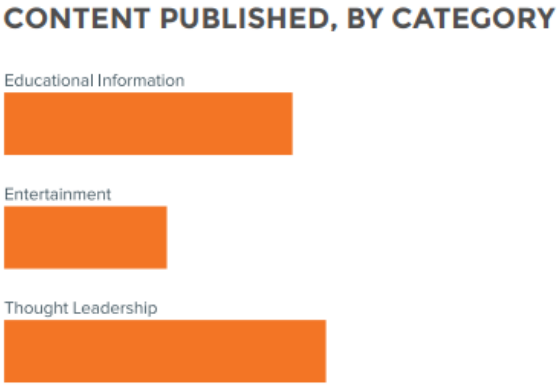
1. *Bar Chart*

   Diagram batang digunakan untuk menunjukkan perubahan dari waktu ke waktu, membandingkan berbagai kategori atau membandingkan bagian-bagian secara keseluruhan. Diagram batang memiliki *style* horizontal dan vertikal dalam penggunaannya.



Gambar 2.2-3 Diagram Batang Vertikal

Style **vertikal** baik digunakan untuk data berurutan / waktu yang berjalan dari kiri ke kanan ataupun memvisualisasikan nilai negatif di bawah sumbu x



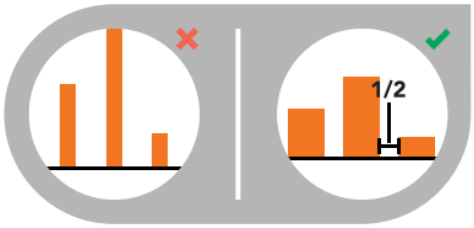
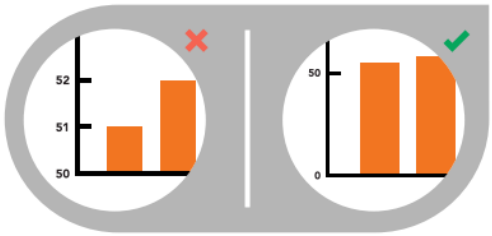
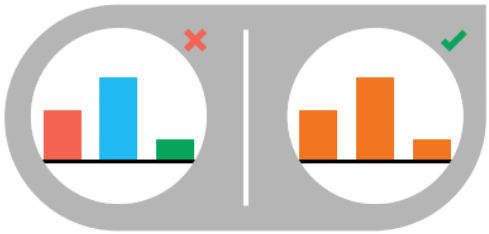
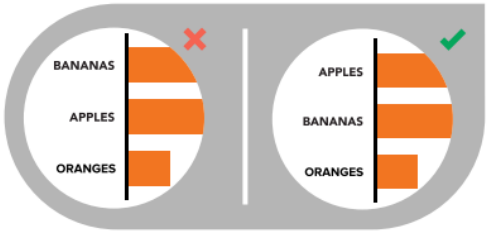
Gambar 2.2-4 Diagram Batang Horizontal

Style **horizontal** baik digunakan untuk data dengan label kategori panjang

Penggunaan diagram batang harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-4.

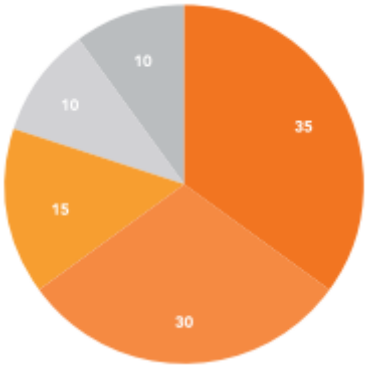
Tabel 2.2-4 Bentuk Desain Diagram Batang

Desain	Keterangan
	Gunakan label horizontal dan hindari penulisan label diagonal atau vertikal untuk diagram batang.

Desain	Keterangan
	Perhatikan penggunaan jarak antara batang harus selebar ½ batang.
	Mulai dari nilai 0 di sumbu Y
	Gunakan warna yang konsisten.
	Urutkan data berdasarkan abjad atau nilai

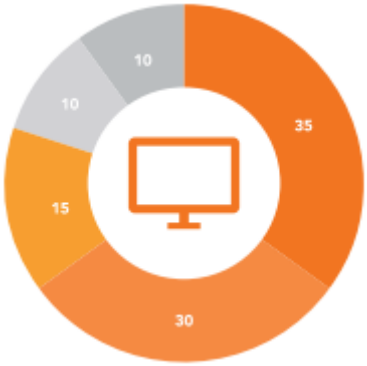
2. *Pie Chart*

   Diagram lingkaran digunakan untuk menggambarkan besar perbandingan setiap bagian dengan total nila keseluruhan baik itu persen. Diagram lingkaran dapat digunakan dengan *style* standar dan donat.



Gambar 2.2-5 Diagram Lingkaran Standar

Style **standar** digunakan untuk menunjukkan hubungan bagian-bagian secara keseluruhan.

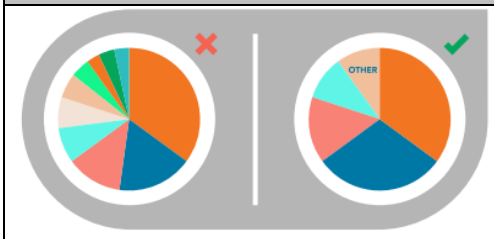


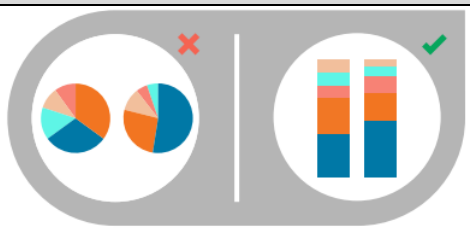
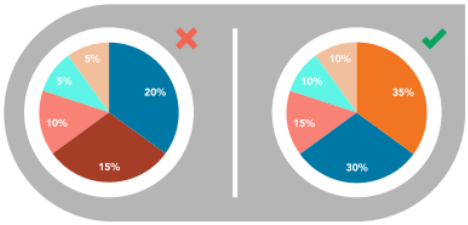
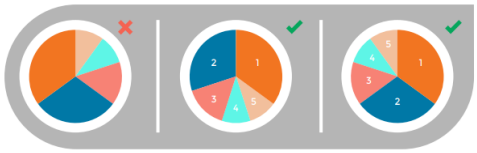
Gambar 2.2-6 Diagram Lingkaran Donat

Style Donat memungkinkan dimasukkannya nilai total atau elemen desain di tengah.

Penggunaan diagram lingkaran harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-5.

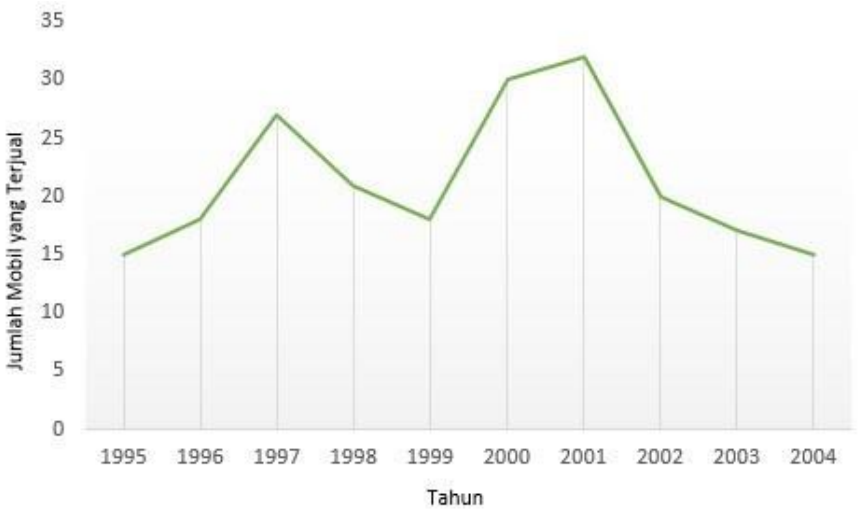
Tabel 2.2-5 Bentuk Desain Diagram Lingkaran

Desain	Keterangan
	• Hindari membuat bagian lebih dari 5 kategori. • Jika kategori lebih dari 5 maka kelompokan nilai yang lebih kecil ke dalam kategori lain-lain, tetapi pastikan hal itu tidak menyembunyikan informasi yang menarik atau signifikan.

Desain	Keterangan
	• Jangan gunakan diagram lingkaran untuk membuat perbandingan. • Jika ada perbandingan gunakan grafik batang bertumpuk
	Pastikan semua data apabila ditambah berjumlah 100% dan ukuran lebar bagian sesuai dengan nilainya.
	Dua cara mengurutkan bagian diagram : 1. Posisikan bagian terbesar pada posisi jam 12, kemudian posisikan bagian terbesar kedua pada posisi jam 12 tetapi berlawan arah. 2. Posisikan bagian terbesar pada jam 12 dengan searah jarum jam. Posisikan bagian yang tersisa dalam urutan menurun

3. *Line Chart*

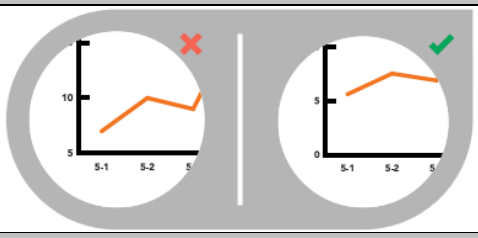
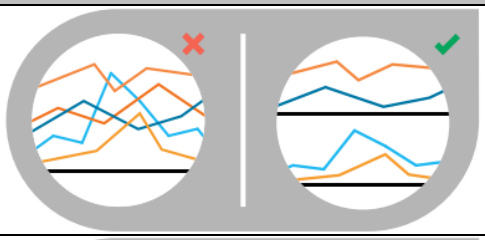
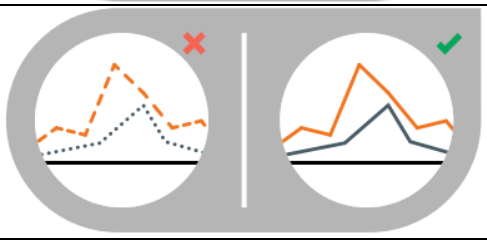
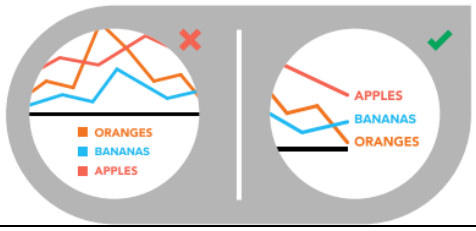
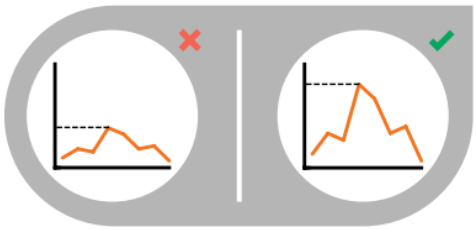
Diagram garis digunakan untuk menunjukkan hubungan deret waktu dengan data kontinu. Diagram garis biasanya menunjukkan tren, akselerasi, perlambatan, volatilitas[18].



Gambar 2.2-7 Periode Penjualan Mobil

Penggunaan diagram garis harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-6.

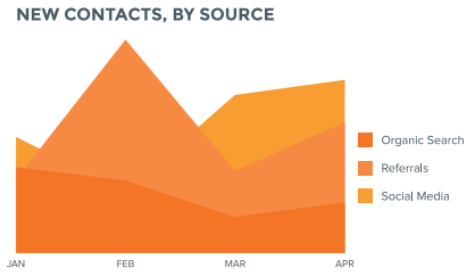
Tabel 2.2-6 Bentuk Desain Diagram Garis

Desain	Keterangan
	Diagram garis dimulai dari awal nol.
Desain	**Keterangan**
	• Kelompok jangan lebih dari 4 baris. • Jika perlu menampilkan lebih banyak, pisahkan dalam diagram terpisah untuk perbandingan lebih baik.
	Gunakan garis solid pada diagram garis.
	Penempatan label di garis secara langsung memungkinkan pembaca dengan cepat mengidentifikasi garis dan label yang sesuai referensi.
	Gunakan tinggi yang tepat

4. *Area Chart*

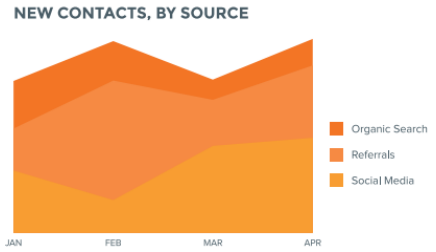
Diagram area digunakan untuk menggambarkan hubungan deret waktu, tetapi mereka berbeda dari diagram garis karena dapat mewakili volume. Diagram

area dapat digunakan dengan *style Area Chart*, *Stacked Area*, dan 100% *Stacked Area.*



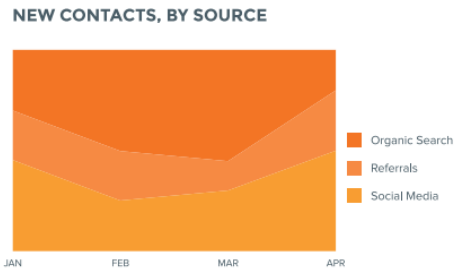
Area Chart digunakan untuk menunjukkan atau membandingkan perkembangan kuantitatif dari waktu ke waktu.

Gambar 2.2-8 Area *Chart*



Stacked Area digunakan untuk memvisualisasikan hubungan bagian dengan keseluruhan dan membantu menunjukkan bagaimana masing-masing kategori berkontribusi terhadap total kumulatif

Gambar 2.2-9 *Stacked* Area

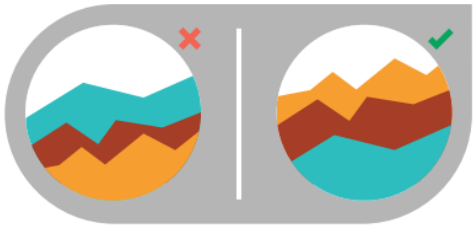
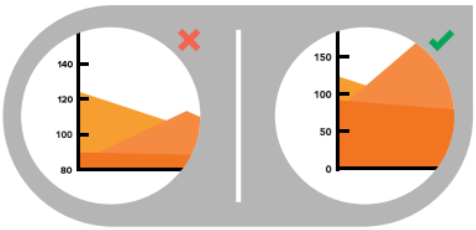
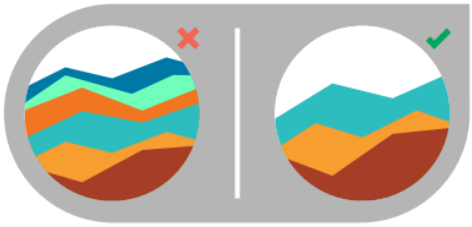
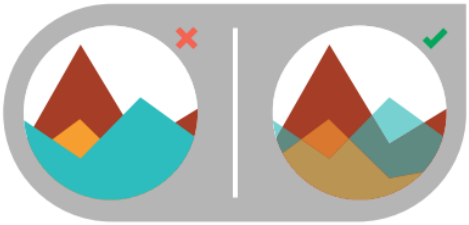
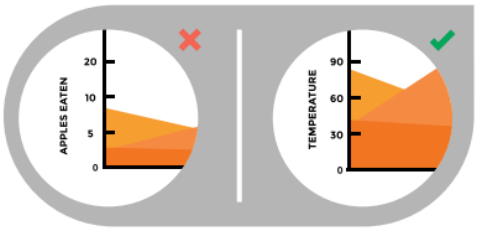


100% *Stacked Area* digunakan untuk menunjukkan distribusi kategori sebagai bagian dari keseluruhan, di mana total kuantitatif tidak penting.

Gambar 2.2-10 100% *Stacked* Area

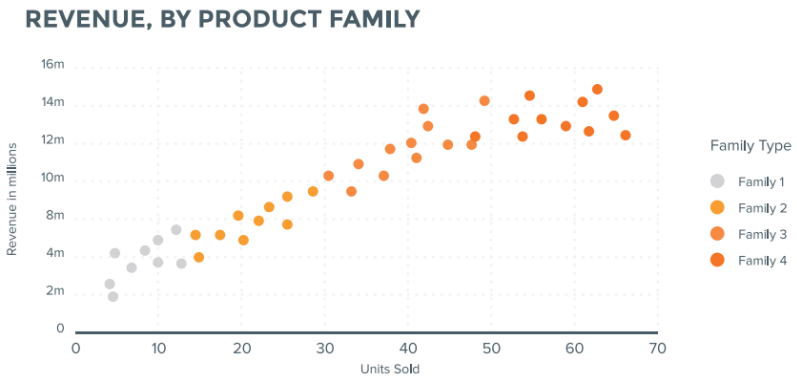
Penggunaan diagram area harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-7.

Tabel 2.2-7 Bentuk Desain Diagram Area

Desain	Keterangan
	Buatlah mudah dibaca Di *Stacked area chart* atur data ke posisi kategori dengan data sangat bervariasi di bagian atas grafik dan variabilitas rendah di bagian bawah.
	Mulai dengan nilai sumbu Y pada nol
	Jangan tampilkan lebih dari 4 kategori data. Jika terlalu banyak akan menghasilkan visual yang berantakan sehingga sulit diuraikan.
	Gunakan warna transparan dalam area *chart* dan pastikan data tidak jelas di latar belakang dengan menggunakan transparansi.
	Jangan gunakan area *chart* untuk menampilkan data *discreate*

5. *Scatter Plot*

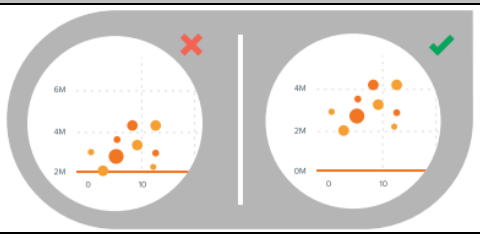
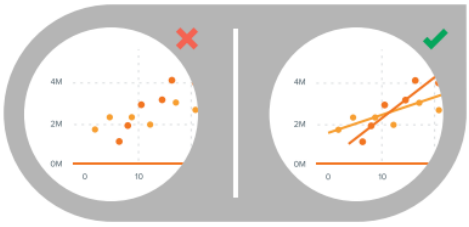
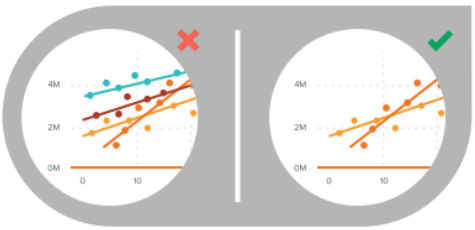
   Scatter plot berfungsi untuk menunjukkan hubungan antara item berdasarkan pada dua set variabel. *Scatter plot* baik digunakan untuk menunjukkan korelasi dalam jumlah besar data.



Gambar 2.2-11 *Scatter* Plot *Revenue Product*

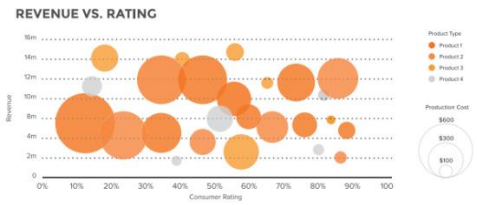
Penggunaan *Scatter Plot* harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-8.

Tabel 2.2-8 Bentuk Desain *Scatter* Plot

Desain	Keterangan
	Mulai dengan nilai sumbu Y pada nol
	Gunakan ukuran dan warna titik untuk merahasiakan variabel data tambahan
	Gunakan garis tren untuk membantu menarik korelasi antara variabel untuk menunjukkan tren.
	Jangan membandingkan lebih dari 2 garis tren karena akan sulit untuk ditafsirkan.

6. Bubble Chart

Bubble chart berfungsi untuk menampilkan perbandingan nominal atau hubungan peringkat. *Bubble chart* dapat digabungkan dengan *Scatter Plot* dan *Head Map*.



Bubble Plot baik digunakan untuk menampilkan variabel tambahan.

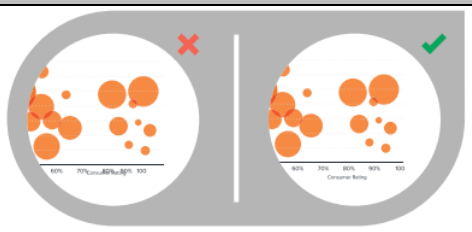
Gambar 2.2-12 *Bubble* Plot

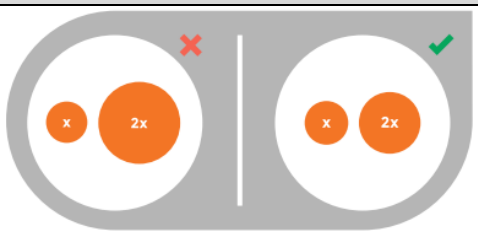
Bubble map baik digunakan untuk memvisualisasikan nilai untuk wilayah geografis tertentu.

Gambar 2.2-13 *Bubble* Map

Penggunaan *Bubble Chart* harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-9.

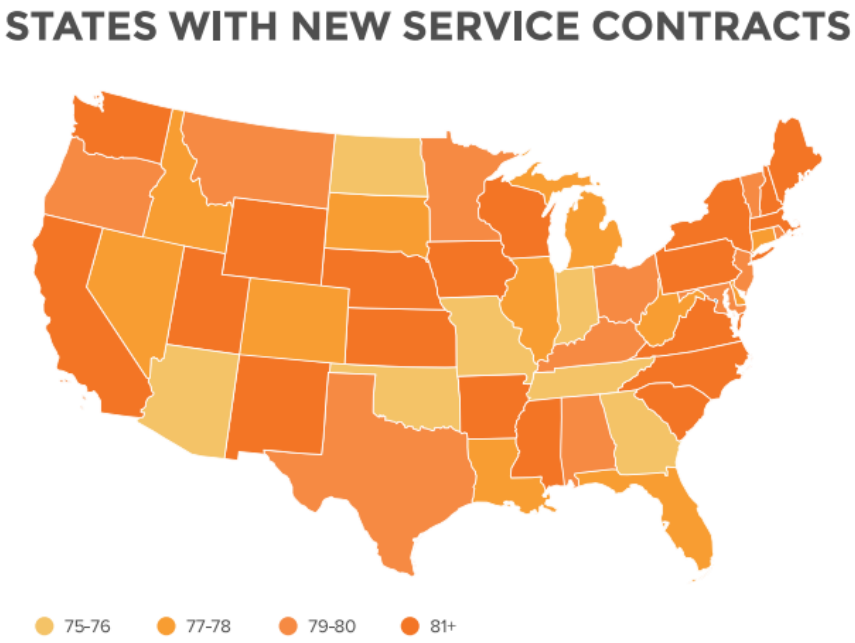
Tabel 2.2-9 Bentuk Desain Bubble Chart

Desain	Keterangan
	Pastikan label bisa terlihat dan mudah untuk diidentifikasi dengan *bubble* yang sesuai.

Desain	Keterangan
	*Bubble* harus diskalakan sesuai area dan bukan diameter.
	Hindari menggunakan bentuk yang tidak sepenuhnya melingkar karena dapat menyebabkan ketidakakuratan.

7. *Heat Map*

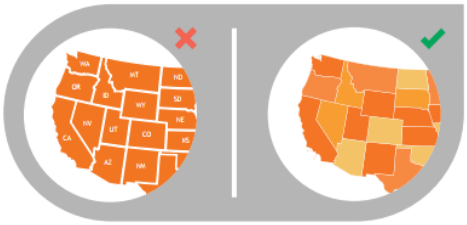
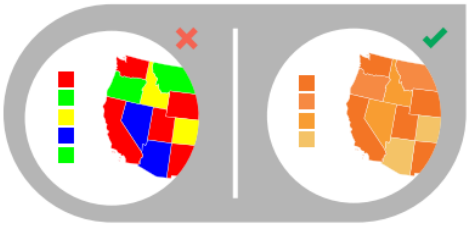
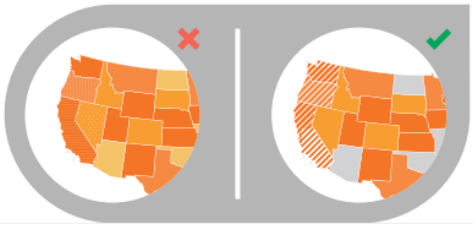
   Heat Map berfungsi untuk menampilkan data *kategorikal* dan menggunakan intensitas warna untuk mewakili nilai area geografis atau tabel data.



Gambar 2.2-14 *Heat* Map New Service *Contract*

Penggunaan *Heat map* harus memperhatikan bentuk desain yang baik dapat dilihat pada Tabel 2.2-10.

Tabel 2.2-10 Bentuk Desain *Heat Maps*

Desain	Keterangan
	Gunakan garis besar peta yang sederhana untuk membingkai data.
	Gunakan warna seperlunya atau satu warna dengan variasi warna antara dua warna utama untuk menunjukkan intensitas.
	Gunakan pola *sparingly* untuk menunjukkan variabel kedua yang dapat di terima.
	Pilih *range* data yang tepat dan gunakan tanda +/- untuk memperluas rentang tinggi dan rendah.

2.2.4.2 Proses Visualisasi Data

Visualisasi data dapat berjalan dengan baik jika penyampaian dari bentuk visualisasi kepada penerima informasi sesuai dengan jawaban yang ingin disampaikan. Membuat bentuk visualisasi data yang baik perlu untuk memahami sebuah data. Proses untuk memahami sebuah data dapat dimulai dari beberapa pertanyaan. Pertanyaan-pertanyaan yang dimulai berdasarkan data dapat di jawab dengan langkah-langkah sebagai berikut[6][17].

1. *Acquire*

   Pada tahap ini digunakan untuk memperoleh data dari berbagai sumber baik dari *file* yang sangat banyak maupun dari jaringan internet.

2. *Parse*

   Pada tahap ini dilakukan penyesuaian data ke format yang telah ditentukan dan di kelompokan ke dalam beberapa kategori agar dapat dengan mudah diketahui jenis-jenisnya.

3. *Filter*

   Pada tahap ini data akan dilakukan proses seleksi (filter) sesuai dengan kebutuhan dan yang tidak sesuai kebutuhan. Data yang tidak sesuai kebutuhan akan dihapus.

4. *Mine*

   Pada tahap ini data akan di implementasikan ke dalam ilmu statistika atau *data mining* untuk melihat pola antar data atau menempatkan data dalam konteks matematika.

5. *Represent*

   Pada tahap ini dilakukan pemilihan model visual dasar yang sesuai dengan data agar mudah dimengerti. Model visual dasar seperti grafik, diagram, garis dan lain-lain.

6. *Refine*

   Pada tahap ini model visualisasi dibuat lebih jelas dan lebih menarik agar dapat mudah dibaca.

7. *Interact*

   Pada tahap ini dilakukan penambahan metode ke sistem untuk memvisualisasikan data agar sesuai keinginan pengguna.

2.2.5 *Usability Testing*

Usability Testing adalah cara untuk mengevaluasi sebuah produk atau jasa dengan cara mengujinya kepada calon pengguna. Umumnya, selama pengujian, pengguna akan mencoba untuk menyelesaikan tugas yang diberikan, sementara pemilik produk akan mengamati, mendengar, dan mencatat temuan[19].

Pengujian ini dilakukan untuk memastikan setiap fungsionalitas yang dijalankan mudah dipahami oleh pengguna. Hal ini dapat dilakukan dengan cara melihat pengguna melakukan tugas yang diberikan, menemukan kesulitan mereka, dan memperbaiki desain sesuai kesulitan pengguna.

Tujuan dari *usability testing* adalah mencari permasalahan yang berkaitan dengan kegunaan, mengumpulkan data kualitatif dan kuantitatif, serta menentukan kepuasan pengguna dengan produk tersebut. *Usability testing* dapat dikelompokkan berdasarkan uji kegunaan seperti *formative testing* dan *summative testing*[20][21].

1. *Formative testing*

   Formative testing merupakan alat pendukung untuk pengambilan keputusan selama tahap awal pada proses desain akan memberikan wawasan berharga tentang di mana pengguna mengalami kesulitan mencapai tujuan.

2. *Summative testing*

   Summative testing adalah alat pendukung untuk pengambilan keputusan setelah pembangunan satu produk dengan tujuan untuk mengetahui satu produk telah memenuhi matriks keberhasilan atau tidak berdasarkan perspektif pengalaman pengguna. Terdapat tiga kriteria utama pada *usability testing* seperti *Effectiveness*, *Efficiency*, *Satisfaction*. Berikut adalah tahapan yang dilakukan sebagai berikut[21].

   a. Menentukan kerangka pengujian

      Tahap pertama dalam *usability testing* adalah menentukan tujuan, hipotesis, dan metode pengujian.

   b. Membuat daftar tugas

      Pada tahap ini dilakukan menentukan tugas yang harus diselesaikan.

   c. Pembuatan skenario pengujian

      Tugas yang telah ditentukan dibuatlah skenario. Skenario berisi peran dari partisipan dan petunjuk yang harus dilakukan partisipan.

   d. Membuat naskah pengujian

Pembuatan naskah ditulis untuk keperluan penelitian seperti pertanyaan, daftar tugas, dan skenario.

e. Melakukan pengujian dan mencatat hasil pengujian

Pada pengujian berlangsung, peneliti harus mencatat setiap hal yang dilakukan oleh partisipan.

f. Melakukan Evaluasi

Pada tahap terakhir dari *usability testing*, dilakukan evaluasi yang bertujuan untuk mendapatkan informasi dan pengetahuan tentang pengujian yang telah dilakukan.

2.2.6 Peningkatan Performance *Database* MYSQL

Performance sebuah *database* merupakan hal yang sangat penting yang harus di perhatikan, karena dengan *performance database* yang baik, akan meringankan kinerja sistem dalam mengakses *query* atau melakukan proses terkait DBMS. Dalam melakukan peningkatan *performance database* terdapat beberapa komponen yang harus diperhatikan saat pembuatan *database*. Komponen yang harus diperhatikan untuk meningkatkan *performance database* yaitu sebagai berikut[22].

1. Optimalisasi dan Pengindeksan Skema

   Melakukan pengoptimalisasi dan pengindeksan skema merupakan salah satu bagian yang dapat meningkatkan kinerja *database*. Membuat perancangan skema yang baik akan menghasilkan kinerja yang tinggi dalam pengeksekusian *database*. Membuat perancangan skema yang baik harus memperhatikan beberapa faktor yaitu sebagai berikut[22].

   a. Pemilihan Jenis Tipe Data

      - Gunakan tipe data terkecil yang dapat menyimpan dan mewakili data yang ingin disimpan.

        Contoh : *Field* Jenis kelamin, tipe data yang tepat adalah **ENUM** dibandingkan menggunakan ***string*** untuk data yang ingin disimpan "**Laki-laki**" dan "**Perempuan**".

- Gunakan tipe data yang lebih sederhana contoh : untuk menyimpan tanggal dan waktu gunakan tipe data bawaan *database* seperti "***datetime***" bukan dengan tipe data "*string*".
- Hindari mendefinisikan kolom tabel NULL jika memungkinkan atau pertimbangkan menggunakan nol, nilai khusus atau *string* kosong.
- Gunakan tipe data yang sama pada tabel terkait(saat *join* pada 2 tabel atau lebih).

b. Penggunaan pengindeksan pada tipe data yang sesuai dengan manfaatnya[23].

a. PRIMARY (*AUTO_INCREMENT*) mempercepat akses data karena mesin penyimpanan tidak perlu memindai seluruh tabel untuk menemukan data yang diinginkan

b. UNIQUE

c. INDEX

d. FULLTEXT

2. Optimalisasi Performance *Query*

Pembuatan struktur *database* yang baik belum cukup untuk mendapatkan kinerja tinggi dalam *database*. Selain pembuatan struktur *database* yang baik dibutuhkan juga untuk merancang *query* yang baik karena optimalisasi indeks, optimalisasi skema dan optimalisasi *query* saling terhubung antara satu dan yang lain.

Kesalahan yang harus dihindari dalam pengeksekusian *query*.

a. Hindari mengambil lebih banyak baris dari yang dibutuhkan.

Contoh : Menampilkan 10 artikel terbaru di halaman depan situs berita.

`SELECT * FROM berita;` → Tidak Direkomendasi

`SELECT * FROM berita LIMIT 10;` → Direkomendasi

b. Mengambil semua kolom dari gabungan multi tabel.

Contoh : Menampilkan semua judul dan sumber berita di situs berita.

`SELECT * FROM berita join detail_sumber ... ;` → Tidak Direkomendasi

`SELECT judul, sumber FROM berita join detail_sumber ... ;` → Direkomendasi

2.2.7 Analisis dan Desain Berorientasi Objek

Analisis dan Desain Berorientasi Objek (*Object Oriented Analysis and Design*) adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep. Dasar pembuatannya sendiri adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Alasan mengapa harus memakai metode berorientasi objek yaitu karena perangkat lunak itu sendiri yang bersifat dinamis, di mana hal ini disebabkan karena kebutuhan pengguna berubah dengan cepat[24].

Selain itu bertujuan untuk menghilangkan kompleksitas transisi antar tahap pada pengembangan perangkat lunak, karena pada pendekatan berorientasi objek, notasi yang digunakan pada tahap analisis perancangan dan implementasi relatif sama tidak seperti pendekatan konvensional yang dikarenakan notasi yang digunakan pada tahap analisanya berbeda-beda. Hal itu menyebabkan transisi antar tahap pengembangan menjadi kompleks[24].

Di samping itu dengan pendekatan berorientasi objek membawa pengguna kepada abstraksi atau istilah yang lebih dekat dengan dunia nyata, karena di dunia nyata itu sendiri yang sering pengguna lihat adalah objeknya bukan fungsinya. Beda ceritanya dengan pendekatan terstruktur yang hanya mendukung abstraksi pada level fungsional. Adapun dalam pemrograman berorientasi objek menekankan berbagai konsep seperti: *Class*, *Object*, *Abstract*, *Encapsulation*, *Polymorphism*, *Inheritance* dan tentunya UML (*Unified Modeling Language*)[24].

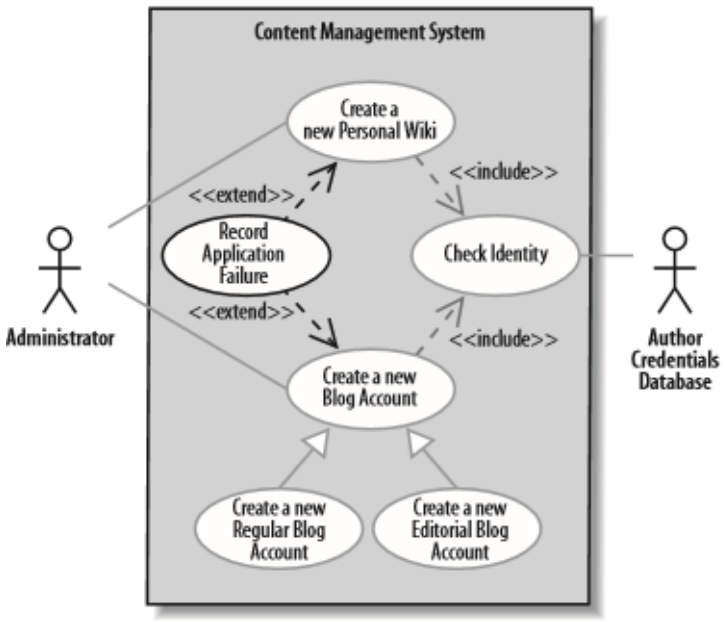
UML (*Unified Modeling Language*) sendiri merupakan salah satu alat bantu yang dapat digunakan dalam Bahasa pemrograman berorientasi objek. Selain itu UML merupakan *standard modeling language* yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bias menyelesaikan tugas-tugas seperti: Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan Testing. Dapat disimpulkan bahwa UML (*Unified Modeling Language*) adalah sebuah Bahasa yang berdasarkan grafik atau gambar untuk memvisualisasikan, melakukan

spesifikasi, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis objek (*Object Oriented Programming*).

Dokumentasi UML menyediakan 10 macam diagram untuk membuat model aplikasi berorientasi objek yang 5 di antaranya sebagai berikut.

1. *Use Case Diagram*

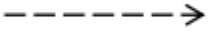
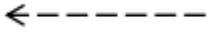
   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Di dalam *use case* diagram ini sendiri lebih ditekankan kepada apa yang diperbuat sistem dan bagaimana sebuah sistem itu bekerja. Sebuah *use case* merepresentasikan sebuah interaksi antara *actor* dengan sistem. *Use case* merupakan bentuk dari sebuah pekerjaan tertentu, misalnya *login* ke dalam sistem, cetak *document* dan sebagainya, sedangkan seorang *actor* adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu[25]. Contoh *use case* dapat dilihat pada Gambar 2.2-15.



Gambar 2.2-15 Contoh *Use Case Diagram*

Simbol yang digunakan pada *use case* diagram dapat dilihat pada Tabel 2.2-11.

Tabel 2.2-11 Simbol pada *use case* diagram

Simbol	Nama Simbol	Keterangan
	*Actor*	Menggambarkan suatu entitas yang berkaitan dengan sistem tapi bukan dari bagian dalam sistem itu sendiri. *Actor* berada di luar sistem namun berkaitan erat dengan fungsionalitas di dalamnya. *Actor* memiliki hubungan secara langsung terhadap fungsi utama baik terhadap salah satu atau semua fungsionalitas utama. *Actor* juga dapat dibagi terhadap berbagai jenis atau tingkatan dengan cara digeneralisasi atau dispesifikasi tergantung kebutuhan sistemnya. *Actor* biasanya dapat berupa pengguna atau *database* yang secara pandang berada dalam suatu ruang lingkup sistem
	*Use case*	Gambaran umum dari fungsi atau proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *use case*, tetapi yang digambarkan hanya fungsionalitas utama yang berkaitan dengan sistem. *Use case* menitik-beratkan bagaimana suatu sistem dapat berinteraksi baik antar sistem maupun di luar sistem.
	Asosiasi	Menghubungkan keterkaitan antara aktor dengan *use case*
	*Include*	Relasi *use case* di mana proses bersangkutan akan dilanjutkan ke proses yang dituju
	*Extend*	Relasi use case tambahan ke sebuah *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu
	*Generalisasi*	Sebuah relasi di mana fungsi yang satu adalah fungsi yang umum dari yang lainnya

2. *Use Case Scenario*

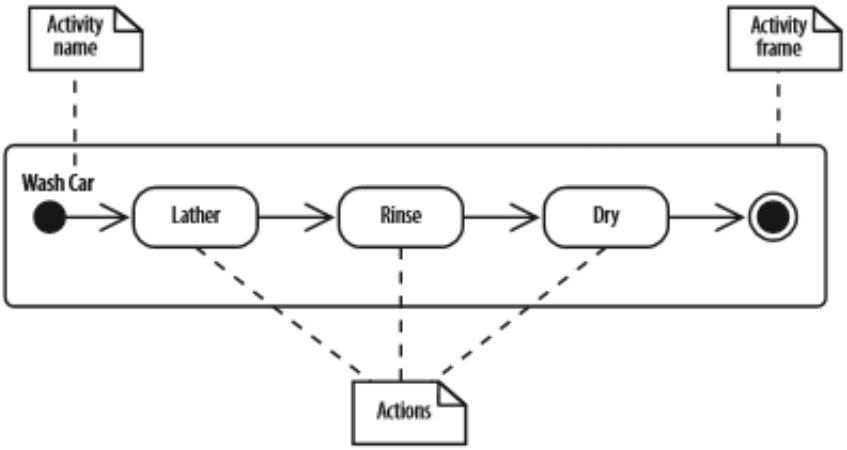
Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per *use case*-nya[25]. [25]. Dasar format penulisan *use case scenario* dapat di lihat pada Tabel 2.2-12.

Tabel 2.2-12 Format Penulisan Use *Case Scenario*

Use Case Name	Berisi nama *use case*	
Related Requirement	Berisi kode kebutuhan yang terkait dengan skenario	
Goal in Context	Tujuan yang ingin dicapai *use case*	
Precondition	Kondisi sistem ketika akan dilangsungkannya skenario *use case*	
Successful End Condition	Kondisi jika *use case* berhasil terpenuhi	
Failed End Condition	Kondisi jika *use case* gagal terpenuhi	
Primary Actor	Aktor utama yang berasosiasi dengan *use case*	
Secondary Actor	Aktor selain aktor utama yang berasosiasi dengan *use case*	
Trigger	Aktivitas yang dilakukan untuk mengawali *use case*	
Included Cases	*Use case* yang di-*include*	
Main Flow	*Step*	*Action*
	1	Langkah-langkah aksi dari aktivitas *use case*
	2	
Extension	*Step*	*Branching Action*
	2.1	Langkah-langkah aksi lain selain aktivitas utama dari *use case*
	2.2	

3. *Activity Diagram*

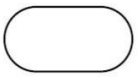
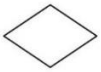
Activity Diagram adalah sebuah tahapan yang lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Di mana biasanya dipakai pada bisnis modeling untuk memperlihatkan urutan aktivitas proses bisnis. *Activity* diagram ini sendiri memiliki struktur yang mirip dengan *flowchart* atau data *flow* diagram pada perancangan terstruktur. *Activity* diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram[25]. Contoh *activity* diagram dapat dilihat pada Gambar 2.2-16.



Gambar 2.2-16 Contoh *Aktivity Diagram*

Simbol yang digunakan dalam *aktivity* diagram dapat dilihat pada Tabel 2.2-13.

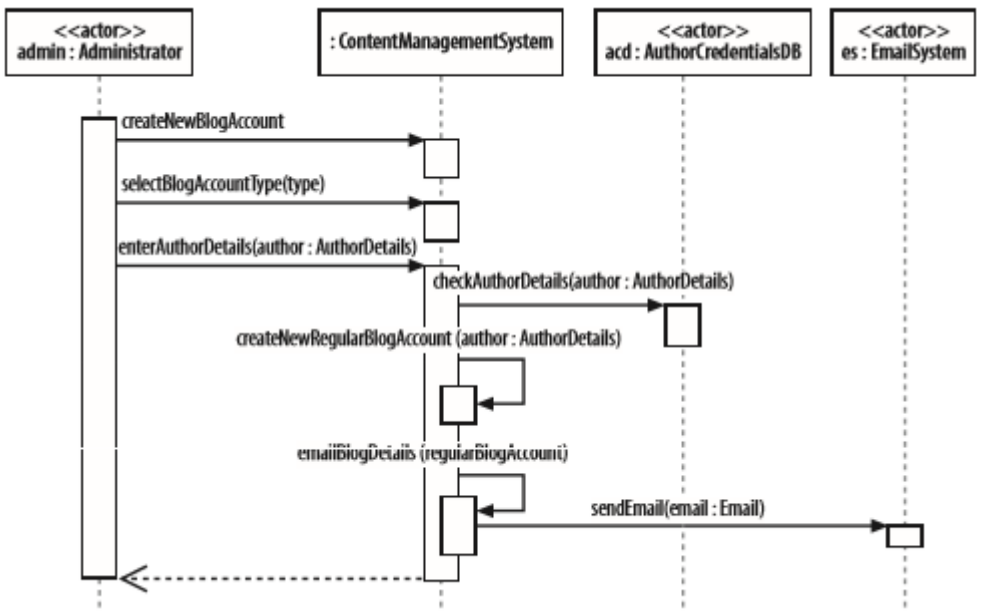
Tabel 2.2-13 Simbol *Aktivity* Diagram

Simbol	Nama Simbol	Keterangan
	*Start State*	Menggambarkan entitas eksternal yang berhubungan dengan sistem
	*Stop State*	Menggambarkan proses yang ada dalam suatu sistem
	*State*	Menggambarkan aktivitas yang terjadi
	*State Condition*	Menerangkan kondisi proses yang sedang berlangsung
	*State Transition*	Menggambarkan perubahan dari *state* satu ke *state* lainnya

4. *Sequence Diagram*

   Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram jenis ini memberikan kejelasan sejumlah objek dan pesan-pesan yang diletakkan di antaranya di dalam sebuah *use case*. Komponen utamanya adalah objek yang digambarkan dengan kotak segi empat atau bulat, *message* yang digambarkan dengan gari putus dan waktu yang ditunjukkan dengan *progress vertical*. Manfaat dari *sequence* diagram adalah

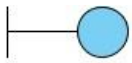
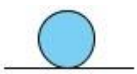
memberikan gambaran detail dari setiap *use case* diagram yang dibuat sebelumnya[25]. Contoh *sequence* diagram dapat dilihat pada Gambar 2.2-17.



Gambar 2.2-17 Contoh *sequence diagram*

Simbol *sequence* diagram dapat dilihat pada Tabel 2.2-14.

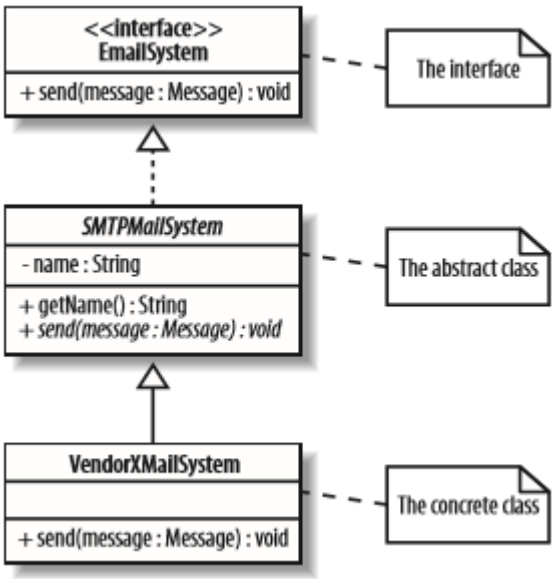
Tabel 2.2-14 Simbol *Sequence* Diagram

Simbol	Nama Simbol	Keterangan
	*Actor*	Menggambarkan pihak yang berhubungan dengan sistem.
	*Object*	Object merupakan *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal. *Object* yang dimaksud dapat berupa *boundary, control,* dan *entity*.
		
		
	LifeLine	Mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah *object*.
	Message	Spesifikasi dari komunikasi antar *object* yang memuat berbagai informasi tentang aktifitas yang terjadi

Simbol	Nama Simbol	Keterangan
	*Activation*	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. Mengindikasikan sebuah *object* yang akan melakukan sebuah aksi.

5. *Class Diagram*

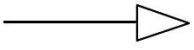
Class diagram adalah sebuah *class* yang menggambarkan struktur dan penjelasan *class*, paket dan objek serta hubungan satu sama lain. *Class* diagram juga menjelaskan hubungan antar *class* secara keseluruhan di dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan[25]. Contoh *class* diagram dapat dilihat pada Gambar 2.2-18.

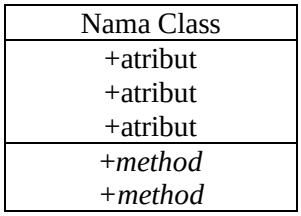
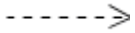
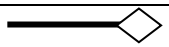


Gambar 2.2-18 Contoh *class diagram*

Simbol *class* diagram dapat dilihat pada Tabel 2.2-15.

Tabel 2.2-15 Simbol Class Diagram

Simbol	Nama Simbol	Keterangan
	*Generalization*	Hubungan di mana objek anak *(descendent)* berbagi perilaku dan struktur data dari *object* yang ada di atasnya *object* induk (*ancestor*).

Simbol	Nama Simbol	Keterangan
	*Class*	*Class* adalah blok - blok pembangun pada pemrograman berorientasi *object*. Sebuah *Class* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan properti atau atribut *class*. Bagian akhir mendefinisikan berbagai *method* dari sebuah *class*.
	*Dependency*	Salah satu jenis relasi *class* diagram yang lemah di mana objek dalam suatu *class* akan bekerja sangat singkat dengan objek yang ada pada *class* lain (*work briefly*)
	Association	Salah satu jenis relasi *class* diagram yang lemah di mana objek dalam suatu *class* akan bekerja dengan jumlah waktu yang lama terhadap objek yang ada pada *class* lain. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum *multiplicity* pada sebuah *relationship*. (Contoh: *One-to-one, one-to-many, many-to-many*).
	*Aggregation*	Jenis relasi *class* di mana ketika satu *class* di *share* atau direferensikan kepada objek yang ada di *class* lain.
	*Composition*	Jenis relasi *class* diagram yang kuat di mana jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class*tempat dia bergantung tersebut. Sebuah relationship *composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.

2.2.8 PHP

PHP adalah salah satu bahasa *Server-side* yang didesain khusus untuk aplikasi web. PHP dapat disisipkan di antara struktur HTML dan karena bahasa *Server side*, maka bahasa PHP akan dieksekusi di server, sehingga dikirimkan ke

browser adalah “hasil jadi” dalam bentuk HTML, dan kode PHP Anda tidak akan terlihat. PHP termasuk dalam bahasa *Open Source*. Jadi Anda dapat mengubah *source code* dan mendistribusikan secara bebas[26].

Adapun kelebihan-kelebihan dari bahasa pemrograman PHP yaitu sebagai berikut.

1. PHP mudah dibuat dan kecepatan akses tinggi.
2. PHP dapat berjalan dalam web server yang berada dan dalam *system* operasi yang berbeda pula. PHP dapat berjalan disistem operasi UNIX, Windows98, Windows NT dan Macintosh.
3. PHP diterbitkan secara gratis
4. PHP juga dapat berjalan pada web server Microsoft Personal Web Server, *Apache*, IIS, *Xitami* dan sebagainya.
5. PHP adalah termasuk bahasa yang *embedded* (bisa ditempel atau diletakkan dalam tag HTML).
6. PHP termasuk *server-side programming*.
7. Sistem *database* yang didukung oleh PHP seperti Oracle, *Sybase*, mSQL, MySQL, Solid, *Generic* ODBC, Postgres SQL.

2.2.9 Laravel

Laravel adalah sebuah *framework* PHP yang dirilis di bawah lisensi MIT, dibangun dengan konsep MVC (*model view controller*). Laravel adalah pengembangan *website* berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan *syntax* yang ekspresif, jelas dan menghemat waktu[27].

MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen- komponen aplikasi, seperti : manipulasi data, *controller*, dan *user interface*.

1. Model → *Model* mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.
2. View → *View* adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web.
3. *Controller* → *Controller* merupakan bagian yang menjembatani model dan *view*.