BAB 2
TINJAUAN PUSTAKA

2.1 Touring

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan kegiatan *touring*, diantaranya sejarah *touring* di Indonesia dan klasifikasi *touring*.

2.1.1 Sejarah *Touring* di Indonesia

Pengenalan *Touring* tidak terlepas dari pengenalan kendaraan bermotor pertama kali di Indonesia. Sepeda motor di Indonesia pertama kali dimiliki oleh seorang berkebangsaan Inggris bernama John C. Potter pada tahun 1893. J.C. Potter dikenal sebagai penjual mobil yang mendapat kepercayaan Sunan Solo untuk mengurusi pengiriman mobil pertamanya dari Eropa.

Seiring dengan pertambahan jumlah kendaraan bermotor yang terus bertambah, lahirlah klub-klub *touring* sepeda motor, yang anggotanya adalah pengusaha perkebunan dan petinggi pabrik gula.

Touring klub motor pertama kali di Indonesia memecahkan rekor perjalanan sejauh 850 Kilometer melintasi Batavia, Bandung, Semarang, Blora, Tjepu menuju Surabaya [13].

2.1.2 Klasifikasi *Touring*

Touring dapat dibagi menjadi beberapa jenis [14], yaitu :

1. *Touring* berdasarkan TEMPAT/WILAYAH
   Domestik Tour, Overseas Tour, Countryside Tour, Intercity Tour dan *Intersland Tour.*
2. *Touring* berdasarkan WAKTU
   Halfday Tour, Fullday Tour, Overnight Tour dan *Multipleday Tour.*
3. *Touring* berdasarkan CARA PENYELENGGARAAN
   Regular Tour dan *Irregular Tour.*
4. *Touring* berdasarkan TUJUAN

Educational Tour, Bussiness Tour, Cultural Tour dan *Convention Tour.*

5. *Touring* berdasarkan SARANA ANGKUTAN
   Overland Tour, Sea Tour, Air Touring dan *Combination.*
6. *Touring* berdasarkan KELAS/TARIF
   Deluxe Tour, First Class Tour, Standar Class Tour, Economy Class Tour dan *Budget Class Tour.*
7. *Touring* berdasarkan JUMLAH PESERTA
   Individual Tour, Grup Tour dan *Mass Tour.*
8. *Touring* berdasarkan MINAT
   General Tour dan *Special Interest Tour.*

2.2 Android

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan sistem operasi Android, diantaranya pengenalan sistem operasi Android, arsitektur Android, dan Android SDK (*Software Development Kit*).

2.2.1 Sistem Operasi Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka [4]. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada masa saat ini, sebagian besar vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis Android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi Android. Hal ini karena

Android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun [4].

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari *Apple* pada sistem operasi *tablet PC*. Pesatnya pertumbuhan Android selain faktor yang disebutkan di atas adalah karena Android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, aplikasi dan *tools* pengembangan, *market* aplikasi Android, serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga Android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia [4].

Android telah melalui cukup banyak pembaruan sejak pertama rilis. Tabel 2.1 menunjukkan berbagai versi Android beserta dengan tanggal rilis dan *codename* (nama kode).

Tabel 2.1 Sistem Operasi Android

Versi Android	Tanggal Rilis	Nama Kode
1.0	23 September 2008	-
1.1	9 February 2009	-
1.5	30 April 2009	*Cupcake*
1.6	15 September 2009	*Donut*
2.0/2.1	26 Oktober 2009	*Éclair*
2.2	20 Mei 2010	*Froyo*
2.3	6 Desember 2010	*Gingerbread*
3.0/3.1/3.2	22 Februari 2011	*Honeycomb*
4.0	19 Oktober 2011	*Ice Cream Sandwich*
4.1/4.2/4.3	9 Juli 2012	*Jelly Bean*
4.4	31 Oktober 2013	*KitKat*
5.0	3 November 2014	*Lollipop*

2.2.2 Arsitetktur Android

Secara garis besar, arsitektur Android terdiri atas 5 bagian, diantaranya sebagai berikut [4]:

1. *Application* dan *Widgets*

   Merupakan *layer* (lapis), dimana pengguna berhubungan dengan aplikasi saja.

2. *Application Frameworks*

   Merupakan *Open Development Platform* yang ditawarkan Android untuk dapat dikembangkan guna membangun aplikasi. Pengembang memiliki akses penuh menuju *Application Programming Interface* (API) *Frameworks* seperti yang di lakukan oleh aplikasi kategori inti. Komponen-komponen yang termasuk di dalam *Applications Frameworks* adalah *Views, Content Provider, Resource Manager, Notification Manager*, dan *Activity Manager*.

3. *Libraries*

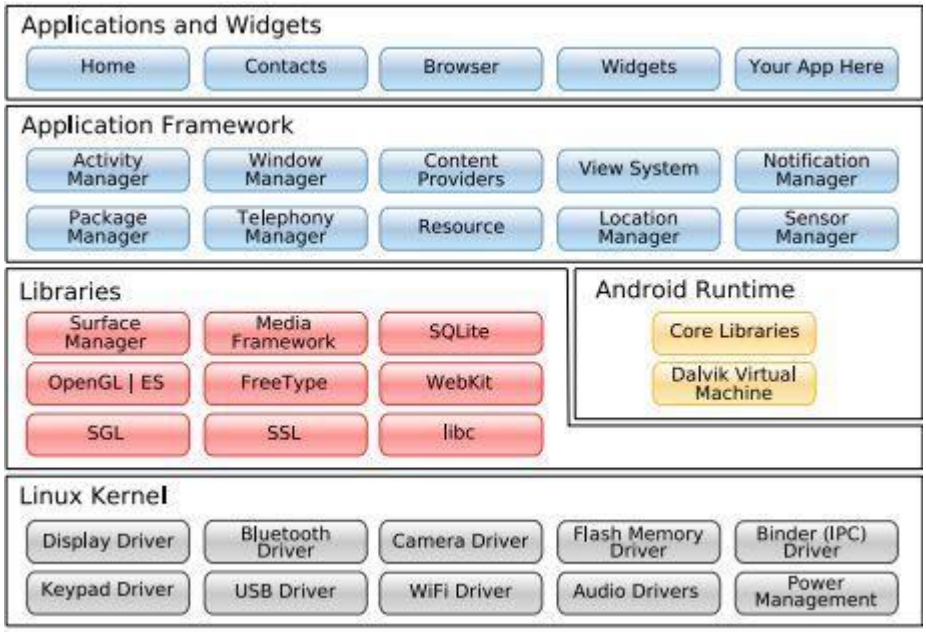
   Merupakan *layer*, dimana fitur-fitur Android berada.

4. Android *Run Time*

   Merupakan *layer* yang membuat aplikasi Android dapat dijalankan, dimana dalam prosesnya menggunakan implementasi Linux.

5. Linux Kernel

   Merupakan *layer* inti dari sistem operasi Android berada.



Gambar 2.1 Arsitekteur Android

2.2.3 Android SDK *(Software Development Kit)*

Android SDK adalah *tools Application Programming Interface* (API) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middeware* dan aplikasi kunci yang dirilis oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan yang bukan merupakan aplikasi bawaan *handphone*/*smartphone* [4].

2.3 *Location Base Service* (LBS)

Location Based Service (LBS) atau layanan berbasis lokasi adalah *service* yang berfungsi untuk mencari dengan teknologi *Global Positioning System* (GPS) dan *Google's cell-based location* [4]. Map dan layanan berbasis lokasi menggunakan lintang (*latitude*) dan bujur (*longitude*) untuk menentukan lokasi geografis, namun sebagai *user* dibutuhkan alamat atau posisi *realtime*, bukan nilai lintang dan bujur [4]. Android menyediakan *geocoder* yang mendukung *forward* dan *reverse geocoding*. Menggunakan *geocoder*, dapat dilakukan konversi nilai lintang bujur menjadi alamat dunia nyata atau sebaliknya.

LBS adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang digunakan. Dua unsur utama dari LBS adalah:

1. *Location Manager (*API *Maps)*

   Menyediakan *tools* atau *source* untuk LBS, *Aplication Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan, memanipulasi *maps* atau peta beserta fitur-fitur lainnya seperti tampilan satelit, *street* 'jalan', maupun gabungannya. Paket ini berada pada *com.google.android.maps* [4].

2. *Location Providers*

   Menyediakan teknologi pencarian lokasi yang digunakan oleh *device* ‘perangkat’. API *Location* berhubungan dengan data GPS *(Global Positioning System)* dan data lokasi *real-time.* API *Location* berada pada paket Android yaitu dalam paket *android location*. Dengan *Location Manager*, dapat ditentukan lokasi saat ini, *track* ‘gerakan/perpindahan’, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan [4].

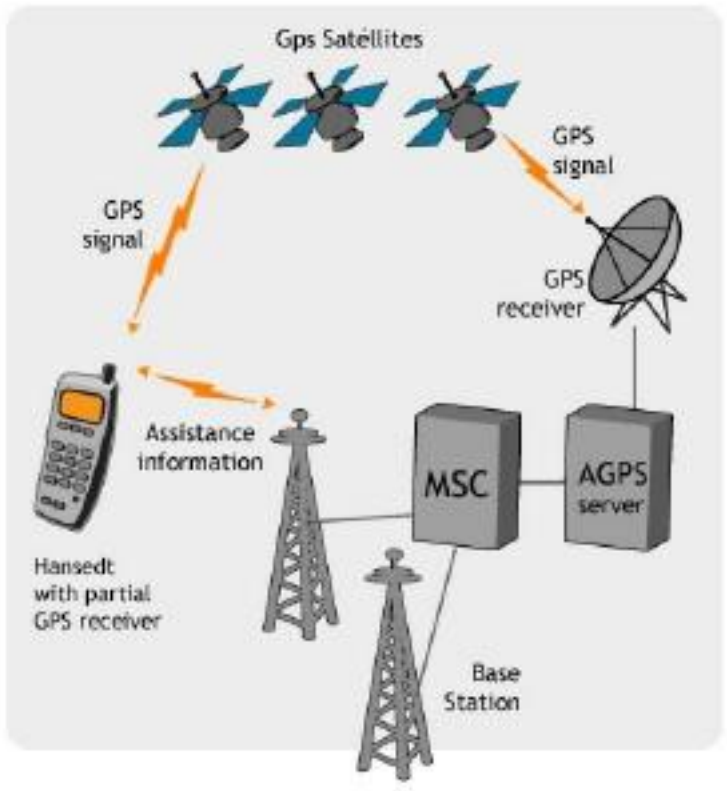
2.4 Assisted – Global Positioning System (A-GPS)

A-GPS merupakan teknologi yang menggunakan *server* bantuan untuk memotong waktu yang dibutuhkan untuk menentukan lokasi penggunaan GPS, sehingga tidak perlu lagi proses komputasi data waktu dari 3 satelit secara langsung di perangkat *Handphone*. Hal ini berguna di daerah perkotaan atau bahkan di dalam ruangan yang banyak penghalang. Teknologi A-GPS saat ini sudah dipasang di semua jenis *smartphone* khususnya android. Namun secara tidak sadar teknologi A-GPS masih disebut dengan GPS. A-GPS menjadi lebih umum disebut dengan layanan berbasis lokasi atau *Location Based Service* (LBS).

Server bantuan penyedia data informasi satelit yang dibutuhkan oleh *system* GPS biasanya di *supply* oleh *Network* operator SIM *card* karena sering kali menara jaringan selular memiliki unit penerima GPS dan unit penerima tersebut secara terus-menerus men-*download* informasi data satelit yang ada di angkasa dan memprosesnya.

Menurut Tanoe (2011), pada sistem A-GPS, telepon genggam akan menangkap sinyal satelit yang lalu dikirimkan ke *server* penyedia layanan telepon, hasil perhitungan lokasi yang dilakukan oleh *server* dikirimkan kembali ke telepon genggam. Peta juga dapat dikirimkan oleh *server* tersebut, atau sudah disimpan pada telepon genggam. Tentunya sistem ini hanya berfungsi bila jaringan telepon genggam mampu dan disediakan bagi pengguna. Oleh karena perhitungan-perhitungan dilakukan oleh *server* penyedia jaringan telepon, maka telepon genggam tidak memerlukan prosesor yang canggih.

Bantuan dari *server* juga memungkinkan telepon genggam untuk mengunci satelit lebih cepat pada saat dinyalakan (TTFF = *time to first fix).* Pada beberapa produk telepon genggam, fasilitas A-GPS ini dapat dimatikan dan digantikan oleh alat lain, misalnya *bluetooth* GPS. Sehingga tidak memerlukan *transfer* data dengan *server operator* telepon.



Gambar 2.2 Cara Kerja A-GPS [15]

2.5 Structured Analysis and Design (SAD)

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan analisis dan perancangan terstruktur, diantaranya pengenalan *Structured Analysis and Design* (SAD) dan *Data Flow Diagram* (DFD).

2.5.1 Pengenalan *Structured Analysis and Design* (SAD)

Analisis terstruktur terfokus pada aliran data dan proses bisnis dan perangkat lunak. Analisis ini disebut *process oriented*. Analisis terstruktur sederhana dalam konsep. Para analisis menggambarkan serangkaian proses dalam bentuk diagram alir data *(data flow diagram)* yang menggambarkan proses yang ada atau yang di usulkan bersama-sama dengan *input*, *output* dan *file* mereka [6].

Setelah tahapan analisis selesai, maka usulan kebutuhan sistem harus diterjemahkan menjadi sistem informasi berbasis komputer. Proses mengubah usulan kebutuhan menjadi perangkat lunak bukan hal yang mudah, harus ada beberapa langkah yang digunakan untuk mempermudah dan menjamin perangkat lunak yang dihasilkan berkualitas. Langkah awal desain biasanya dimulai dengan pemodelan sistem. Model digunakan untuk menyederhanakan cara mengkomunikasikan proses-proses bisnis yang harus dilakukan sistem dengan cara yang formal antar pemain pengembangan sistem.

Pemodelan proses adalah cara formal untuk menggambarkan bagaimana bisnis beroperasi. Mengilustrasikan aktivitas-aktivitas yang dilakukan dan bagaimana data berpindah di antara aktivitas-aktivitas itu. Ada banyak cara untuk mempresentasikan proses model. Cara yang paling populer adalah dengan menggunakan *Data Flow Diagram* (DFD) [6].

2.5.2 Data Flow Diagram (DFD)

Suatu diagram arus data atau *Data Flow Diagram* (DFD) adalah penyajian grafis dari sebuah sistem yang mempergunakan empat bentuk simbol untuk mengilustrasikan bagaimana data mengalir melalui proses-proses yang saling tersambung [6].

Ada empat elemen yang menyusun suatu DFD, yaitu:

1. Proses

   Aktivitas atau fungsi yang dilakukan untuk alasan bisnis yang spesifik, biasa berupa manual maupun terkomputerisasi.

2. *Data Flow*

   Satu data tunggal atau kumpulan logis suatu data, selalu diawali atau berakhir pada suatu proses.

3. *Data Store*

   Kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam *data store*. Aliran data di-*update* atau ditambahkan ke *data store*.

4. *External Entity*

   Orang, organisasi, atau sistem yang berada di luar sistem tetapi berinteraksi dengan sistem.

DFD terdiri dari beberapa level, diantaranya sebagai berikut:

1. *Context Diagram*

   Context Diagram (Diagram Konteks) merupakan DFD pertama dalam proses bisnis. Menunjukkan dimana proses bisnis berada. Menunjukkan semua proses bisnis dalam 1 proses tunggal (proses 0). *Context* diagram juga menunjukkan semua entitas luar yang menerima informasi dari atau memberikan informasi ke sistem.

2. *Level 0 Diagram*

   Menunjukkan semua proses utama yang menyusun keseluruhan sistem. Level ini juga menunjukkan komponen internal dari proses 0 dan menunjukkan bagaimana proses-proses utama direlasikan menggunakan *data flow*. Pada *level* ini juga ditunjukkan bagaimana proses-proses utama terhubung dengan entitas eksternal. Pada *level* ini juga dilakukan penambahan *data store*.

3. *Level 1 Diagram*

   Umumnya diagram *level* 1 diciptakan dari setiap proses utama dari *level* 0. *Level* ini menunjukkan proses-proses *internal* yang menyusun setiap proses-proses utama dalam *level* 0, sekaligus menunjukkan bagaimana informasi berpindah dari satu proses ke proses yang lainnya. Jika misalnya proses induk

dipecah, katakanlah menjadi 3 proses anak, maka 3 proses anak ini secara utuh menyusun proses induk.

4. *Level 2 Diagram*

   Menunjukkan semua proses yang menyusun sebuah proses pada *level* 1. Bisa saja penyusunan DFD tidak mencapai *level* 2 ini. Atau mungkin harus dilanjutkan ke *level* berikutnya (*level* 3, *level* 4, dan seterusnya).

2.6 Object Oriented Analysis and Design (OOAD)

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan analisis dan perancangan berorientasi objek, diantaranya pengenalan *Object-Oriented Analysis and Design* (OOAD) dan *Unified Modelling Language* (UML).

2.6.1 Pengenalan *Object Oriented Analysis and Design* (OOAD)

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemrograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme), dan *inheritance* (pewarisan) [7].

Analisis dan desain berorientasi objek atau *Object-Oriented Analysis and Design* (OOAD) adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, menyiapkan dokumentasi, serta perancangan program dan basis data. Pertama-tama, suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya [7].

Analisis Berorientasi Objek atau *Object-Oriented Analysis* (OOA) dimulai dengan menyatakan suatu masalah, analisis membuat model situasi dari dunia nyata, menggambarkan sifat yang penting. Dalam menganalisa suatu sistem, analisis harus bekerja dengan pihak yang membutuhkan sistem untuk memahami masalah tersebut dengan jelas. Model analisis adalah abstraksi yang ringkas dan tepat dari apa yang harus dilakukan oleh sistem dan bagaimana melakukannya. Objek dalam model harus merupakan konsep domain dari aplikasi dan bukan merupakan implementasi komputer seperti struktur data. Model yang baik harus dipahami dan ditanggapi oleh ahli aplikasi. Empat kesulitan yang merupakan gangguan utama dalam pengembangan sistem adalah memahami *problem* domain, komunikasi antara pihak yang berkaitan, perubahan kontinyu, dan *reuse* (penggunaan kembali) [7].

Desain Berorientasi Objek atau *Object-Oriented Design* (OOD) merupakan tahap lanjutan setelah Analisis Berorientasi Objek dimana tujuan sistem diorganisasi ke dalam *sub*-sistem berdasar struktur analisis dan arsitektur yang dibutuhkan. Desainer sistem (*system designer*) menentukan karakteristik penampilan secara optimal, menentukan strategi memecahkan masalah, dan menentukan pilihan alokasi sumber daya. Sebagai contoh, desainer sistem mungkin menentukan perubahan pada *screen* untuk *workstation* yang memerlukan kecepatan serta resolusi lebih tinggi [7].

Desain model digunakan berdasarkan model analisis dengan dilengkapi rincian untuk implementasi. Fokus dari desain objek (*object design*) adalah perencanaan struktur data dan algoritma yang diperlukan untuk implementasi setiap kelas. Objek domain aplikasi dan objek domain komputer dijelaskan dengan menggunakan konsep dan notasi berorientasi objek yang sama [7].

Kelas, objek dan relasi dikembangkan dalam tahap pembuatan desain objek yang pada akhirnya diterjemahkan ke dalam bahasa pemrograman, basis data, dan implementasi perangkat keras. Hal yang penting dalam tahap implementasi adalah mengikuti penggunaan perangkat lunak yang baik [7].

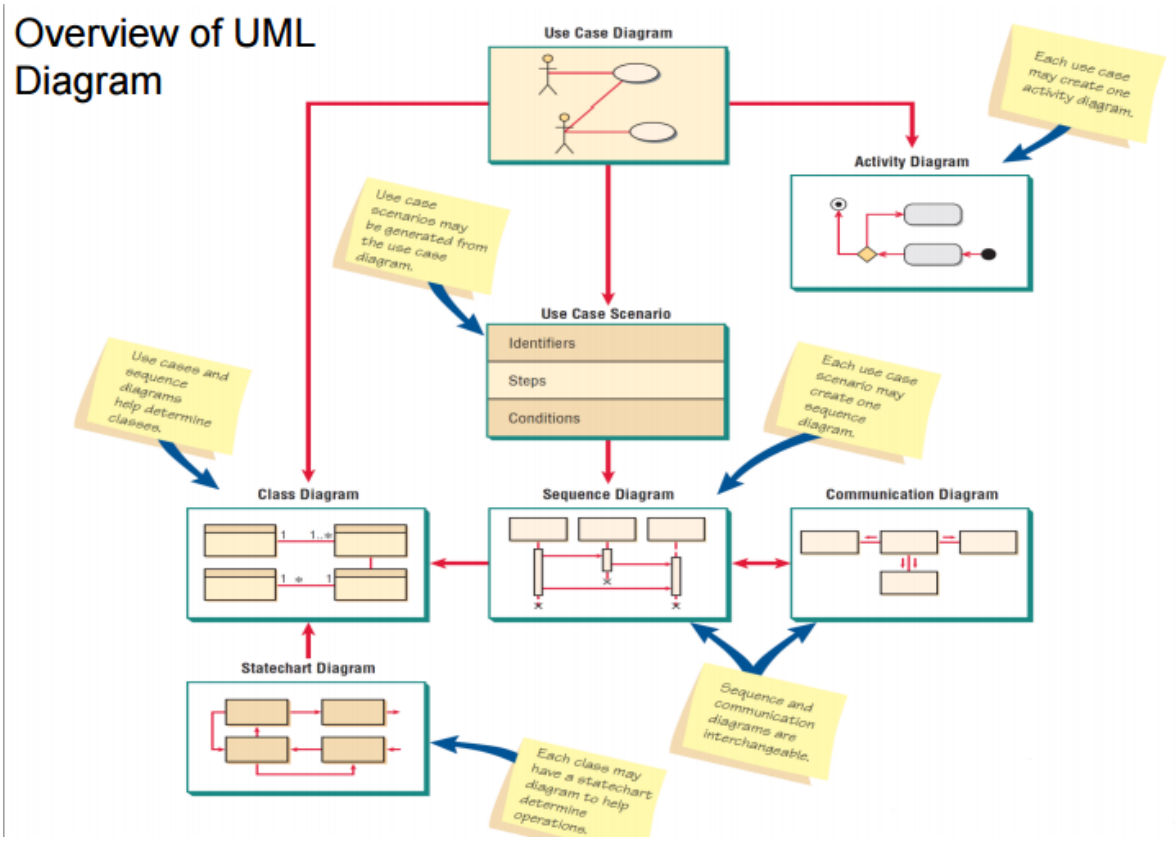
2.6.2 Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak *level* abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, serta didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*) [7].

Tujuan utama perancangan UML adalah:

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.
2. Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep inti.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep-konsep pengembangan *level* lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern*.

UML menyediakan diagram-diagram yang sangat kaya dan dapat diperluas sesuai kebutuhan kita. Diagram adalah representasi secara grafis dari elemen-elemen tertentu beserta hubungan-hubungannya. Diagram penting karena diagram menyediakan representasi secara grafis dari sistem (atau bagiannya). Representasi grafis sangat memudahkan pemahaman terhadap sistem [7]. Diagram-diagram yang dikunakan pada UML dapat dilihat pada gambar 2.3 [17].



Gambar 2.3 Diagram UML

Diagram di dalam UML terbagi menjadi dua, diantaranya:

1. Diagram Struktur

   Diagram ini memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan aspek statik dari sistem. Diagram struktur di UML terdiri dari:

   a. Diagram Kelas (*Class Diagram*)

      Diagram ini menunjukkan sekumpulan kelas, *interface*, serta kolaborasi dan keterhubungannya. Diagram kelas ditujukan untuk pandangan statik terhadap sistem.

b. Diagram Objek (*Object Diagram*)

Diagram ini menunjukkan sekumpulan objek dan keterhubungannya. Diagram ini menunjukkan potongan statik dari instan-instan yang ada di diagram kelas. Diagram ini untuk memperlihatkan satu prototipe atau kasus tertentu yang mungkin terjadi.

c. Diagram Komponen (*Component Diagram*)

Diagram ini menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini merupakan pandangan statik terhadap implementasi sistem.

d. Diagram *Deployment* (*Deployment Diagram*)

Diagram ini menunjukan konfigurasi pemrosesan saat jalan dan komponen-komponen yang terdapat di dalamnya. Diagram ini merupakan pandangan statik dari arsitektur.

2. Diagram Perilaku

Diagram perilaku adalah diagram yang sifatnya dinamis (cenderung berubah). Terdapat beberapa jenis diagram yang termasuk kategori diagram perilaku, diantaranya:

a. Diagram *Use Case* (*Use Case Diagram*)

Diagram ini menunjukkan sekumpulan kasus fungsional dan aktor (jenis kelas khusus) dan keterhubungannya.

b. Diagram Sekuen (*Sequence Diagram*)

Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan-pesan yang terjadi.

c. Diagram Kolaborasi (*Collaboration Diagram*)

Diagram ini merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek-objek yang mengirim dan menerima pesan.

d. Diagram *Statechart* (*Statechart Diagram*)

Diagram ini adalah *state-machine* diagram, berisi *state*, transisi, kejadian, dan aktivitas. *Statechart* merupakan pandangan dinamis dari sistem. Diagram ini penting dalam memodelkan perilaku antarmuka, kelas, kolaborasi, dan menekankan pada urutan kejadian. Penting untuk sistem reaktif yang dipicu kejadian di dunia nyata.

e. Diagram Aktivitas (*Activity Diagram*)

Diagram ini menunjukan aliran aktivitas di sistem. Diagram ini adalah pandangan dinamis terhadap sistem. Diagram ini penting untuk memodelkan fungsi sistem dan menekankan pada aliran kendali di antara objek-objek.

2.7 Web Services

Perangkat peranti lunak integrasi aplikasi perusahaan sifatnya spesifik terhadap suatu produk, artinya perangkat ini hanya dapat bekerja dengan bagian-bagian peranti lunak dan sistem operasi tertentu. Contoh, suatu perangkat EAI untuk menghubungkan bagian tertentu dari peranti lunak pencatatan penjualan untuk aplikasi produksi, pengangkutan, dan penagihan, tidak dapat bekerja dengan peranti lunak pencatatan penjualan dari vendor lainnya. Program *middleware* yang dikembangkan oleh sistem BEA, sebuah integrator yang besar, tidak dapat berhubungan dengan aplikasi *middleware* vendor lainnya yang dibeli pada tahun-tahun sebelumnya tanpa pengeluaran yang besar untuk pemrograman dan perancangan. Layanan *web* (*web services*) mencoba memberikan alternatif standar untuk menghadapi masalah-masalah integrasi seperti itu dengan menciptakan sebuah lingkungan komunikasi yang netral [4].

Layanan *web* (*web services*) adalah serangkaian komponen peranti lunak yang bertukar informasi antara satu sama lain dengan bebas menggunakan komunikasi *web* dan bahasa yang standar. Layanan *web* dapat bertukar informasi antara dua sistem yang berbeda, terlepas dari sistem operasi atau bahasa pemrograman sistem tersebut. Layanan *web* dapat dipadukan untuk membangun sistem penghubung aplikasi berbasis *web* standar terbuka dari dua organisasi yang berbeda, dan juga dapat

digunakan untuk membuat aplikasi yang menghubungkan sistem terpisah dalam satu perusahaan. Layanan *web* tidak terikat dengan sistem operasi atau bahasa pemrograman apa pun, dan aplikasi-aplikasi yang berbeda dapat digunakan untuk berkomunikasi satu sama lain dengan cara standar tanpa penulisan kode khusus yang menghabiskan waktu [4].

2.8 Java Script Object Notation (JSON)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemprograman *Java Script*, Standar ECMA-262 Edisi ke-3 Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan lain-lain. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data [5].

JSON terbuat dari dua struktur, diantaranya sebagai berikut:

a. Kumpulan pasangan nama atau nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*structure*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
b. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini [5].

JSON menggunakan bentuk sebagai berikut:

1. Objek (*Object*)

   Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik (*Array*)

   Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai (*value*)

   Nilai dapat berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. *String*

   String adalah kumpulan dari nol atau lebih karakter *unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan *string* C atau Java.

5. Angka

   Angka pada JSON sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.

2.9 Basis Data

Kata basis data atau *database* biasa digunakan untuk menguraikan segala sesuatu dari sekumpulan data tunggal, seperti daftar telepon. Istilah basis data tidak termasuk aplikasi, yang terdiri dari form dan *report* dimana pengguna akan saling berhubungan. Basis data terdiri dari *file-file* fisik yang ditetapkan berdasarkan komputer saat menerapkan perangkat lunak basis data. Di sisi lain, suatu model basis

data lebih kepada konsep dibandingkan objek fisik dan digunakan untuk menciptakan tabel dalam basis data [12].

Basis data adalah suatu aplikasi terpisah yang menyimpan suatu koleksi data. Masing-masing basis data memiliki satu API atau lebih yang berbeda untuk menciptakan, mengakses, mengelola, mencari, dan mereplikasi data [12].

Sebuah basis data adalah tempat penyimpanan *file* data. Sebagai *file* data, suatu basis data tidak menyajikan informasi secara langsung kepada pengguna. Pengguna harus menjalankan aplikasi untuk mengakses data dari basis data dan menyajikannya dalam bentuk yang bisa dimengerti. Ketika bekerja dengan *file-file* data, suatu aplikasi harus dikodekan agar bekerja dengan struktur masing-masing *file* data. Biasanya, suatu basis data berisi suatu katalog yang menggunakan aplikasi untuk menentukan cara data diorganisir. Aplikasi basis data umum bisa menggunakan katalog tersebut untuk menampilkan data dengan pengguna dari basis data yang berbeda secara dinamis, tanpa terikat pada format data tertentu [12].

2.9.1 Database Management System (DBMS)

Database Management System (DBMS) atau sistem manajemen basis data adalah suatu sistem perangkat lunak kompleks yang mengatur permintaan dan penyimpanan data ke dan dari *hard disk*. DBMS menyediakan keamanan (*security*), privasi (*privacy*), integritas (*integrity*), dan kontrol konkurensi (*concurrency controls*). DBMS mengelola transaksi pada *multi user*, lingkungan akses bersamaan, dan menyediakan tingkat independensi data yang mengisolasi pandangan (*view*) pengguna atau aplikasi dari perubahan yang berlangsung di tingkat internal dan konseptual [12].

DBMS adalah perangkat lunak yang disajikan oleh penjual basis data. Produk perangkat lunak seperti *Microsoft Access, Oracle, Microsoft SQL Server, Sybase, DB2,* INGRES, dan MySQL adalah DBMS [12].

DBMS menyediakan semua layanan dasar yang diperlukan untuk mengorganisir dan memelihara basis data, termasuk layanan berikut [12]:

1. Memindahkan data ke dan dari *file-file* data fisik jika dibutuhkan.

2. Mengelola akses data oleh berbagai pengguna secara bersamaan; mencakup ketentuan untuk mencegah peng-*update*-an secara bersamaan.
3. Mengelola transaksi sehingga perubahan transaksi basis data adalah semua atau tidak sama sekali. Dengan kata lain, jika transaksi berhasil, semua basis data yang dibuat berubah dan direkam di dalam basis data. Jika transaksi gagal, tidak satupun dari perubahan tersebut yang direkam oleh basis data.
4. Mendukung bahasa *query* (*query language*), yang mana suatu sistem perintah mempekerjakan pengguna basis data untuk mendapatkan data kembali dari basis data.
5. Ketentuan untuk mem-*backup* basis data dan pemulihan dari kegagalan.

Mekanisme keamanan untuk mencegah perubahan dan akses data yang tidak sah.

2.9.2 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau yang dikenal dengan DBMS (*Database Management System*), *database* ini *multi thread* dan *multi user* [10].

Kekuatan MySQL tidak ditopang oleh sebuah komunitas, seperti *Apache*, yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh pemilik masing-masing, tetapi MySQL didukung penuh oleh sebuah perusahaan profesional dan komersial, yakni MySQL AB dari Swedia [10].

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis [10].

Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan *database server* lainnya, terutama dalam kecepatan. Berikut ini beberapa keistimewaan MySQL, antara lain [10]:

1. *Probability*

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

2. *Multi user*

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

3. *Security*

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

4. *Scalability* dan *limits*

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

2.9.3 Model Entity Relationship

Model *Entity-Relationship* (ER) mula-mula diusulkan oleh Peter Chen pada tahun 1976 sebagai cara untuk mempersatukan pandangan basis data jaringan dan relasional. Langkah sederhana dari model ER adalah model data dan konseptual yang memandang dunia nyata sebagai kesatuan (*entities*) dan hubungan (*relationships*). Komponen dasar model merupakan diagram *entity-relationship* yang digunakan untuk menyajikan objek data secara visual. Chen menulis bahwa model telah diperluas dan kini bisa digunakan untuk merancang basis data. Keunggulan model ER dalam perancangan tersebut adalah [12]:

1. Mampu memetakan model relasional dengan baik. Pembangunan yang digunakan di dalam model ER dengan mudah diubah ke dalam tabel relasional.

2. Sederhana dan mudah dipahami hanya dengan sedikit pelatihan. Oleh karena itu, model bisa digunakan oleh perancang basis data untuk mengkomunikasikan perancangan kepada pengguna akhir.
3. Sebagai tambahan, model bisa digunakan sebagai suatu rencana perancangan oleh pengembang basis data untuk menerapkan suatu model data dalam perangkat lunak manajemen basis data spesifik.

Perancangan basis data dengan menggunakan model *entity-relationship* adalah dengan menggunakan *Entity Relationship Diagram* (ERD). ERD mengilustrasikan struktur logis dari basis data.

2.9.4 Model Relasional

Model relasional secara formal diperkenalkan oleh Dr. E. F. Codd pada tahun 1970 dan berkembang sejak saat itu melalui serangkaian tulisan. Model relasional menyajikan data berbentuk tabel dua dimensi (*two-dimension*). Masing-masing tabel menyajikan beberapa orang, tempat, sesuatu hal, atau peristiwa sebagai informasi yang dikumpulkan. Suatu basis data relasional adalah suatu kumpulan tabel *two-dimensional* [12].

Model relasional merupakan model data yang paling banyak digunakan saat ini. Tingkatan-tingkatan data yang menyusun *database* disebut skema, dimana setiap tingkatan menggambarkan sebuah data. Model relasional ini memfokuskan pada pembahasan hubungan (relasional) antar data. Skema relasi pada model relasional ditentukan oleh nama skema, nama, dan tipe *field* atau kolom. Model data relasional merepresentasikan datanya menggunakan bentuk tabel-tabel yang saling berelasi/berhubungan dengan suatu *field* kunci. Setiap tabel memiliki banyak kolom dengan nama yang unik untuk setiap kolomnya. Model relasi menempati *level* abstraksi yang lebih rendah daripada model ER, karena sifatnya langsung menunjukkan bentuk *record* yang akan disimpan dalam suatu *file* [12].

2.10 Pemrograman *Web*

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan pemrograman web, diantaranya *Hyper Text Markup Language* (HTML), PHP, *Cascade Style Sheet* (CSS) dan JavaScript.

2.10.1 *Hyper Text Marup Language* (HTML)

HTML adalah singkatan dari *Hyper Text Markup Language*. HTML digunakan untuk membuat halaman *web*. Sebuah *file* dokumen yang ditulis dalam format HTML akan dibaca dan diterjemahkan oleh *web browser* (misal Internet *Explorer*) untuk kemudian disajikan dalam bentuk *web* [11].

File-file yang ditulis dalam format HTML disimpan dengan ekstensi *.htm* atau *.html*. *File-file* tersebut dapat ditulis atau diketik menggunakan berbagai macam teks editor, misalkan Notepad, Wordpad, dan lain sebagainya [11].

HTML terdiri atas berbagai macam *tag* yang digunakan untuk menandai dan mengatur tampilan halaman *web* yang dibuat. Misalkan jika ingin membuat sebuah kalimat bercetak tebal di *web*, maka gunakan *tag* yang cocok untuk itu [11].

Tag biasanya dituliskan berpasangan di awal dan di akhir bagian *file* yang akan ditandai. Untuk *tag* yang diletakkan di awal bagian ditulis dengan tanda lebih kecil (<), nama *tag*, dan tanda lebih besar (>). Dan untuk *tag* yang diletakkan di akhir bagian ditulis dengan tanda lebih kecil (<), garis miring (/), nama *tag*, dan tanda lebih besar (>) [11].

2.10.2 PHP

PHP adalah singkatan dari PHP: *Hypertext Preprocessor*. PHP merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting* [10]. PHP memungkinkan untuk membuat halaman *web* yang bersifat dinamis.

PHP dapat dijalankan pada berbagai macam sistem operasi, misalnya Windows, Linux, dan Mac OS. Selain Apache, PHP juga mendukung beberapa *web server* lain, misalnya Microsoft IIS, Caudium, PWS, dan lain-lain [10].

PHP dapat memanfaatkan *database* untuk menghasilkan halaman *web* yang dinamis [10]. Sistem manajemen *database* yang sering digunakan bersama PHP

adalah MySQL. Namun, PHP juga mendukung sistem manajemen *database* Oracle, Microsoft Access, Interbase, dBase, PostgreSQL, dan lain-lain [10].

Hingga kini, PHP sudah berkembang hingga versi 5. PHP 5 mendukung penuh *Object-Oriented Programming* (OOP), integrasi XML, mendukung semua ekstensi terbaru MySQL, pengembangan *web services* dengan SOAP dan REST, serta ratusan peningkatan lainnya dibandingkan versi sebelumnya. PHP juga bersifat *open source* sehingga setiap orang dapat menggunakannya secara gratis [10].

2.10.3 *Cascading Style Sheet* (CSS)

CSS (*Cascading Style Sheet*) adalah suatu teknologi yang digunakan untuk memperindah tampilan halaman *website* (situs). Singkatnya dengan menggunakan metode CSS ini dapat dengan mudah mengubah secara keseluruhan sekaligus memformat ulang situs. CSS dapat digunakan untuk menghindari pekerjaan berulang-ulang [11].

CSS mempunyai dua bagian utama yaitu *selector* dan deklarasi. Yang disebut *selector* biasanya elemen html yang ingin di ubah, sedangkan deklarasi biasanya terdiri dari *property* dan nilai, *property* sendiri adalah atribut *style* yang ingin di ubah, dan setiap *property* memiliki nilai. Setiap *syntax* css selalu diakhiri dengan titik koma (;), dan kelompok deklarasi dikelilingi oleh kurung kurawal ({}) [11].

Ada dua cara untuk membuat aturan *style*, yaitu dengan menuliskannya langsung dalam dokumen HTML atau dengan memisahkannya dalam sebuah *file* khusus yang berekstensi *.css*. Cara yang pertama dapat dilakukan dengan menambahkan *tag* <STYLE> diantara *tag* <HEAD> dan </HEAD>. Kemudian tuliskan aturan *style*-nya di dalam *tag* <STYLE> tersebut. Cara pertama sudah jarang dilakukan. Saat ini, para pembuat *web* biasanya menggunakan cara yang kedua, yaitu membuat aturan *style* dalam *file* terpisah [11].

2.10.4 JavaScript

JavaScript merupakan bahasa *scripting* (bahasa pemrograman yang ringan) yang popular di Internet dan dapat bekerja di sebagian besar *browser* popular seperti

Internet *Explorer* (IE), *Mozilla FireFox*, *Netscape*, *Google Chrome*, dan *Opera*. Kode *JavaScript* dapat disisipkan dalam halaman *web* menggunakan *tag* <SCRIPT> [11].

JavaScript adalah bahasa interpreter yang berarti skrip dieksekusi tanpa proses kompilasi [11]. JavaScript berisi baris kode yang dijalankan di komputer (*web browser*). *JavaScript* didesain untuk menambah interaktif suatu *web* [11].

Ada beberapa cara untuk menuliskan JavaScript, yaitu di bagian <HEAD>, <BODY> atau *external*. JavaScript dalam sebuah halaman *web* akan dieksekusi saat halaman di-*load* oleh *browser*. Namun, hal ini kadang tidak sesuai dengan keinginan karena suatu saat akan mengeksekusi skrip saat halaman di-*load*, tetapi kadang *JavaScript* dijalankan melalui *trigger* sebuah *event* [11].

2.11 Pemrograman Java

Pada bagian ini dijelaskan hal-hal yang berkaitan dengan pemrograman java, diantaranya pengenalan bahasa pemrograman Java dan Android Studio.

2.11.1 Bahasa Pemrograman Java

Java merupakan bahasa pemrograman yang menanjak popularitasnya dewasa ini. Selain gratis, Java mudah didapatkan dan juga tangguh. Java sendiri lahir pada tahun 1991 yang diprakarsai oleh tim Sun Microsystems melalui proyek bernama *Green* yang dipimpin oleh James Gosling [8].

Java menerapkan fitur-fitur dari bahasa pemrograman lain, dimana fitur itu dianggap merupakan siatu kelebihan oleh tim Sun [8]. Misalnya JVM/JRE yang merupakan mesin maya pada bahasa Pascal, Sintaks, dan *Exception Handling* diambil dari C/C++ dan lain sebagainya.

Java Virtual Machine (JVM) atau *Java Runtime Environment* (JRE) merupakan fitur Java yang membuatnya dapat berjalan pada semua *platform* sistem operasi. Apabila membuat Java pada Windows, dapat pula dijalankan pada sistem operasi Linux, Mac OS, dan lainnya [8].

Java memiliki beberapa keunggulan yang tidak dimiliki oleh bahasa lain [8]. Berdasarkan *white paper* resmi dari Sun Microsystems, Java memiliki karakteristik sebagai berikut[8]:

1. Sederhana

   Bahasa pemrograman Java menggunakan sintaks mirip C++, namun pada Java telah banyak diperbaiki terutama menghilangkan penggunaan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.

2. Berorientasi objek (*object oriented*)

   Java menggunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali.

3. Dapat didistribusi dengan mudah

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. *Interpreter*

   Program Java dijalankan menggunakan *intrepeter*, yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.

5. Robust

   Java mempunyai reliabilitas yang tinggi. *Compiler* pada Java mempunyai kemampuan mendeteksi *error* secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi *error* pada pemrograman.

6. Aman

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

   Pemrograman Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada *platform* yang berbeda dengan *Java Virtual Machine*.

8. Portabel

   Source code maupun program Java dapat dengan mudah dibawa ke *platform* yang berbeda-beda tanpa harus dikompilasi ulang.

9. *Performance*

   Performance pada Java sering dikatakan kurang tinggi. Namun *performance* Java dapat ditingkatkan menggunakan kompilasi Java lain seperti Inprise, Microsoft, ataupun Symantec yang menggunakan *Just In Time Compilers* (JIT).

10. *Multithread*

    Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. Dinamis

    Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu *class* dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

2.11.2 Android Studio

Android studio merupakan sebuah *Integrated Development Environment* (IDE) untuk *platform* Android. Android Studio ini diumumkan pada tanggal 16 Mei 2013 pada Konferensi GoogleI/O oleh Produk Manajer Google, Ellie Powers. Android studio bersifat *free* dibawah Apache License 2.0. Android Studio awalnya dimulai dengan versi 0.1 pada bulan mei 2013, Kemudiandibuat versi beta 0.8 yang dirilis pada bulan juni 2014. Yang paling stabil dirilis pada bulan Desember 2014, dimulai dari versi 1.0. Berbasiskan JetBrainns' IntelliJ IDEA, Studio didesain khusus untuk Android *Development*.

Berikut ini beberapa keunggulan yang terdapat pada Android Studio :

1. *Live Layout : WYSIWYG Editor – Live Coding – Real time App Rendering*
2. *Developer Console : optimization tips, assistance for translation, referral tracking,campaigning and promotions – Usage Metrics*
3. *Provision for beta releases and staged rollout*
4. *Gradle-based build support*

5. *Android – specific refactoring and quick fixes*
6. *Lint tools to catch performance, usability, version compatibility and other problems*
7. *ProGuard and app-signing capabilities*
8. *Template-based wizards to create common Android designs and components*
9. *A rich layout editor that allows user to drag-and-drop UI components, option to previewlayouts on multiple screen configurations*
10. *Support for building Android Wear apps*
11. *Built-in support for Google Cloud Platform, enabling integration with Google Cloud Messaging and App Engine.*

2.12 Pengujian *Black Box*

Pengujian *black box* fokus pada keperluan penelusuran kesalahan fungsional dari *software*. Ujicoba *black box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan *interface.*
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan performa.
5. Kesalahan inisialisasi dan terminasi.

Teknik pengujian *blackbox* terdiri dari 10 jenis diantaranya *Equivalence Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing.* Salah satunya yang akan dibahas adalah *Equivalence partitioning*.

Equivalence partioning merupakan metode ujicoba *black box* yang membagi domain *input* dari program menjadi beberapa kelas data dari kasus ujicoba yang dihasilkan. Kasus uji penanganan *single* yang ideal menemukan sejumlah kesalahan (misalnya: kesalahan pemrosesan dari seluruh data karakter) yang merupakan syarat lain dari suatu kasus yang dieksekusi sebelum kesalahan umum diamati.

2.13 Pengujian Kuisioner

Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian ia diminta untuk menilai pernyataan-pernyataan tersebut.

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (STS), Tidak Setuju (TS), Ragu-Ragu (R), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel 2.2.

Tabel 2.2 Pengolahan Kuisioner

Jenis Pernyataan	**Bobot pendapat**				
	SS	**S**	**R**	**TS**	**STS**
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu, totalkan seluruh bobot jawaban tersebut kemudian bagi dengan total responden yang nantinya menjadi nilai rata-rata. Nilai rata-rata inilah yang diambil sebagai acuan sikap dimana jika nilai rata-rata kurang dari 3, maka dapat diartikan responden bersikap negatif dan jika nilai rata-rata lebih dari sama dengan 3, maka dapat diartikan responden bersikap positif terhadap tujuan yang ingin dicapai. Untuk lebih jelasnya dapat dilihat pada rumus dibawah ini [16].

$$\overline{X} = \frac{\sum Total}{n}$$

Dimana,

$\overline{X} \geq 3$ *bersikap positif*

$\overline{X} < 3$ *bersikap negatif*

Keterangan:

$\overline{X}$ = rata-rata

$\sum Total$ = jumlah seluruh nilai setelah dikalikan dengan bobot

n = total responden