BAB 2
LANDASAN TEORI

2.1. Tinjauan Pustaka

Tinjauan pustaka merupakan penjelasan gambaran dari teori-teori kepustakaan yang bersifat relevan terhadap penelitian yang akan dibangun, dengan menulusuri kepustakaan dari berbagai sumber yang bisa didapatkan dari jurnal, buku, *e-book*, atau yang lain sebagainya. Tinjauan pustaka yang akan dibahasa yaitu pengertian Konveksi, Sistem, dan Metode yang digunakan dan bahasa pemrograman yang digunakan dalam pembangunan aplikasi.

2.2. Konveksi

Konveksi adalah suatu industri kecil skala rumah tangga yang merupakan tempat produksi kaos, sweater, jaket, dan lain sebagainya yang pembuatannya dalam jumlah banyak dengan berdasarkan ukuran standar yang telah ditentukan oleh pelanggan, dan proses pengerjaannya dilakukan oleh satu orang dengan secara menyeluruh mulai dari pemotongan bahan, jahit, sampai proses pakaian itu jadi.

Karena produk yang dihasilkan oleh industri konveksi termasuk kedalam salah satu kebutuhan dasar manusia, kepopuleran bisnis konveksi ini akan terus mengalami peningkatan. Didukung oleh permintaan pasar yang begitu besar peluang untuk memulai dan mengembangkan usaha konveksi sangat besar, karena makin banyaknya model-model *fashion* yang sedang mengalami perkembangan. Dalam industri konveksi, proses untuk pembuatan sebuah pakaian disebut dengan *cut*, *make*, dan *trim* [1]. Dan pengertiannya sebagai berikut :

1) *Cutting* : pembuatan pola atau *patron*, *marker*, *cutting*, dan *numbering* [1].
2) *Making* : menjahit dari awal mulai dari bahan kain yang sudah dipola hingga pakaian atau baju siap pakai [1].
3) *Trimming* : *washing* atau dyeing, membersihkan benang, *ironing* atau setrika, penempelan label, dan *packing* pada pakaian atau baju yang sudah jadi [1].

Proses produksi yang dalam usaha konveksi dilakukan secara menyeluruh dan dilakukan oleh setiap operator jahit, mulai dari penjahit kerah, lengan, dan seterusnya sampai menjadi satu pakaian utuh. Berbeda halnya dengan proses

produksi pakaian di pabrik garment yang hanya dilakukan berdasarkan jenis prosenya. Di Kota Bandung sendiri perusahaan konveksi sangat banyak dan berbagai macam jenis konveksi yang disuguhkan mulai dengan jenis konveksi yang hanya menerima produksi baju seragam sekolah, kemeja, kaos, jaket, dan lain sebagainya.

Dengan beragamnya jenis konveksi ini terkadang membuat pelanggan selalu mengalami kebingungan dan kesulitan dalam mencari suatu tempat konveksi yang tepat dan sesuai dengan kebutuhannya. Untuk mempermudah pelanggan dalam mencari tempat konveksi ini maka muncul sebuah ide untuk membuat atau membangun sebuah aplikasi yang bisa membantu pelanggan dalam merekomendasikan suatu tempat konveksi yang tepat dan sesuai dengan kebutuhan pelanggan inginkan.

2.3. Sistem

Sistem adalah sekumpulan komponen atau elemen yang saling berkaitan satu sama lain dalam melakukan kegiatan untuk memudahkan aliran suatu informasi atau materi untuk menghasilkan keluaran yang sudah direncanakan.

2.3.1.Syarat-syarat Sistem

Pada pembangunan suatu sistem harus memenuhi beberapa persyaratan, diantaranya :

1. Sistem harus dibentuk untuk menyelesaikan suatu masalah.
2. Elemen sistem harus mempunyai rencana yang akan ditetapkan.
3. Adanya keterkaitan dari setiap elemen sistem.
4. Unsur dari proses (arus informasi, energi dan material) lebih penting dari pada elemen sistem.
5. Tujuan organisasi lebih penting dari pada tujuan elemen.

2.3.2.Elemen Sistem

Pada sebuah sistem memiliki beberapa elemen, yaitu : tujuan, masukan, proses, keluaran, batas, mekanisme pengendalian dan umpan balik serta lingkungan. Berikut merupakan penjelasan mengenai elemen-elemen yang membentuk sebuah sistem :

1. Tujuan

2. Masukan
3. Proses
4. Keluaran
5. Batas
6. Mekanisme Pengendalian dan Umpan Balik
7. Lingkungan

2.3.3.Klasifikasi Sistem

Klasifikasi sistem merupakan suatu komponen yang memiliki kesatuan yang sama antara komponen lainnya, dengan tujuan sistem memiliki tujuan akhir yang berbeda untuk setiap kasus atau perkara yang terjadi dalam setiap sistem. Maka sistem tersebut dapat diklasifikasikan menjadi beberapa sistem, diantaranya :

1. *Deterministik Sistem*

   Sistem operasi (input atau output) yang didalamnya dapat ditentukan atau diketahui dengan pasti.

2. *Probabilistik Sistem*

   Sistem input dan prosesnya dapat didefinisikan, tetapi output yang dihasilkan tidak dapat ditentukan dengan pasti, sistem ini memiliki sedikit kesalahan atau penyimpangan terhadap ramalan suatu jalannya sistem.

3. *Open Sistem*

   Sistem yang dapat menyesuaikan diri dengan lingkungannya sehingga dapat meneruskan eksistensinya.

4. *Closed Sistem*

   Sistem yang prosesnya tidak mengalami pertukaran materi,energi atau informasi dengan lingkungan di luar sistem.

5. *Relatively Closed Sistem*

   Sistem yang bersifat tertutup tetapi dalam pengoperasiannya dapat menerima pengaruh dari luar yang sudah didefinisikan dalam batasan tertentu.

6. *Artificial Sistem*

   Sistem yang dapat menyerupai kejadian alam atau dapat menirukan kejadian-kejadian yang ada di alam.

7. *Natural Sistem*

   Sistem yang dibentuk dari kejadian dalam alam.

8. *Manned Sistem*

   Sistem yang menjelaskan tingkah laku manusia dalam keikutsertaannya.

2.4. Metode Rekomendasi

Metode rekomendasi merupakan rangkaian cara, pendekatan, atau proses untuk memberikan suatu saran yang bisa diterapkan pada pembangunan aplikasi yang akan dibangun dan bisa sesuai dengan keinginan pelanggan. Oleh karena itu metode rekomendasi membutuhkan model rekomendasi yang tepat agar rekomendasi yang diberikan bisa sesuai dan memudahkan pelanggan untuk mengambil keputusan yang tepat dalam menentukan konveksi yang akan dipilih. Ada beberapa metode atau teknik yang digunakan dalam rekomendasi. Setiap metode disesuaikan dengan permasalahan dalam menghasilkan suatu informasi yang sesuai.

Metode yang dipakai pada permasalahan ini menggunakan metode rekomendasi *Multi-Criteria Decision Making* (MCDM) dengan metode pemecahan permasalahnya menggunakan metode *Weighted Product Method* (WP).

2.4.1.Multi-Criteria Decision Making (MCDM)

Multiple Criteria Decision Making (MCDM) menurut Kusumadewi (2006) adalah suatu metode pengambilan keputusan dalam menetapkan suatu alternatif terbaik dari sejumlah alternatif berdasarkan beberapa kriteria tertentu. Kriteria biasanya berupa ukuran-ukuran, aturan-aturan atau standar yang digunakan dalam pengambilan keputusan [2].

Secara umum dapat dikatakan bahwa MCDM menyeleksi terbaik dari sejumlah *alternative*. Janko (2005) dalam Kusumadewi et al, (2006) menyebutkan terdapat beberapa fitur umum yang digunakan dalam MCDM, yaitu:

1. Alternatif, merupakan obyek-obyek yang berbeda dan memiliki kesempatan yang sama untuk dipilih oleh pengambil keputusan.
2. Atribut, atribut sering juga disebut sebagai kriteria keputusan.
3. Konflik antara kriteria, beberapa kriteria biasanya mempunyai konflik antara satu dengan yang lainnya, misalnya kriteria keuntungan akan mengalami konflik dengan kriteria biaya.
4. Bobot keputusan menunjukkan kepentingan relatif dari setiap kriteria, W = (w1, w2, w3, .., wn).
5. Matriks keputusan, suatu matriks keputusan X yang berukuran m x n, berisi elemen-elemen xij yang merepresentasikan rating dari alternatif Ai ; I = 1, 2, 3, …, m terhadap kriteria Cj ; j = 1, 2, 3, … , n.

Ada beberapa metode yang dapat digunakan untuk menyelesaikan masalah MCDM, antara lain:

1) *Weighted Product Method* (WP)
2) *Simple Additive Weighting Method* (SAW)
3) ELECTRE
4) *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)
5) *Analytic Hierarchy Process*(AHP)
6) *Simple Multi Attribute Rating Assessment* (ARAS)

Metode yang digunakan dari metode *Multiple Criteria Decision Making* ini menggunakan metode penyelesaian dengan *Weighted Product Method* (WP), karena sesuai dengan kasus yang telah dianalisis sebelumnya yaitu untuk mencari solusi dalam rekomendasi tempat konveksi yang sesuai dengan kebutuhan pelanggan. Karena dalam hal produksi pakaian setiap orang mempunyai kriteria sendiri dalam memilih jenis pakaian yang akan mereka produksi. Jika suatu tempat konveksi diberi rating yang bagus oleh seseorang belum tentu sesuai untuk pelanggan yang lainnya.

Dalam MCDM ada 2 komponen penting yang perlu diperhatikan, yaitu : *alternative* dan *criterias*, dua hal itulah yang menyebabkan MCDM menjadi

solusi yang tepat untuk digunakan dalam rekomendasi konveksi. Berikut merupakan penjelasan singkat *Weighted Product Method* dengan perhitungan dalam pencarian solusinya, dan contoh soal dalam metode WP.

2.4.1.1 Weighted Product Method (WP)

Metode WP ini merupakan metode penyelesaian pada sistem pendukung keputusan. Metode ini mengevaluasi beberapa alternatif terhadap sekumpulan atribut atau kriteria, dimana setiap atribut saling tidak bergantungan satu dengan yang lainnya [2]. Konsep dasar metode WP ini menggunakan perkalian untuk menghubungkan rating atribut, dimana rating setiap atribut harus dipangkatkan terlebih dahulu dengan bobot atribut yang bersangkutan (Yoon, 1989). Langkah-langkah penyelesaian metode WP sebagai berikut:

1. Menentukan kriteria-kriteri, yaitu menentukan kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu Ci dan sifat dari masing-masing kriteria [3].
2. Menentukan rating kecocokan, yaitu kecocokan setiap alternatif pada kriteria, dan buat matriks keputusan [3].
3. Melakukan normalisasi bobot [3].
4. Bobot Ternormalisasi = Bobot setiap kriteria/ penjumlahan semua bobot kriteria. Nilai dari total harus memenuhi persamaan :

$$\sum_{j=1}^{n} w_j = 1.$$

Dimana $\sum W_j$ adalah pangkat bernilai positif untuk atribut biaya [3].

5. Menentukan nilai vektor S [3]:

$$S_i = \prod_{j=1}^{n} x_{ij}^{w_j} \;,\; \text{i=1,2,...,m}$$

Keterangan :

S : Menyatakan preferensi alternatif yang dianalogikan sebagai vektor S

x : Menyatakan nilai kriteria

w : Menyatakan bobot kriteria

i : Menyatakan alternative

j : Menyatakan kriteria

n : Menyatakan banyak kriteria

6. Menentukan nilai vektor V

Yaitu nilai yang akan digunakan untuk perangkingan. Nilai preferensi relatif dari setiap alternatif dapat dihitung dengan rumus [3]:

$$V_i = \frac{\prod_{j=1}^{n} x_{ij}^{w_j}}{\prod_{j=1}^{n} (x_j)^{w_j}} ; \quad \text{i=1,2,..,m}$$

Keterangan :

V : Menyatakan preferensi alternatif yang dianalogikan sebagai vektor V

x : Menyatakan nilai kriteria

w : Menyatakan bobot kriteria

i : Menyatakan alternative

j : Menyatakan kriteria

n : Menyatakan banyak kriteria

Merangking Nilai Vektor V, sebagai kesimpulan untuk tahap akhir [3].

Contoh Kasus :

Pada tahun 2010 yang lalu Andika akan kuliah dan kakaknya berjanji untuk membelikan sepeda motor sesuai yang diinginkan andika. Andika memiliki pilihan yaitu motor ninja (M1), tiger (M2), vixsion (M3). Andika memiliki 5 kriteria dalam pemilihan sepeda motor yang nantinya akan dibeli yaitu:

C1 : Desain yang bagus

C2 : Berkualitas

C3 : Irit bahan bakar

C4 : Harga cukup terjangkau

C5 : Warna netral

Kriteria keuntungan : C1, C2, C5

Kriteria biaya : C3, C4

Rating kecocokan setiap alternatif pada setiap kriteria, dinilai dengan 1 sampai 5, yaitu :

1 = Sangat Buruk

2 = Buruk

3 = Cukup

4 = Baik

5 = Sangat Baik

Misal untuk pengambilan keputusan bobot preferensinya sebagai berikut :

$$W = (5, 4, 4, 3, 3)$$

Membuat tabel rating kecocokan :

PRODUK	KRITERIA				
	C1	C2	C3	C4	C5
M1	5	4	3	2	3
M2	4	4	3	2	3
M3	3	3	4	3	3

Bobot awal W = (5, 4, 4, 3, 3)

W1 = 5 / 5+4+4+3+3 = 5/19 = 0.2631

W2 = 4 / 5+4+4+3+3 = 5/19 = 0.2105

W3 = 4 / 5+4+4+3+3 = 5/19 = 0.2105

W4 = 3 / 5+4+4+3+3 = 5/19 = 0.1578

W5 = 3 / 5+4+4+3+3 = 5/19 = 0.1578

Penyelesaian :

Menghitung vektor s

$S_1 = (5^{0.2631}) (4^{0.2105}) (3^{-0.2105}) (2^{-0.1578}) (3^{0.1578}) = 1.7300$

$S_2 = (4^{0.2631}) (4^{0.2105}) (3^{-0.2105}) (2^{-0.1578}) (3^{0.1578}) = 1.6313$

$S_3 = (3^{0.2631}) (3^{0.2105}) (4^{-0.2105}) (3^{-0.1578}) (3^{0.1578}) = 1.2568$

Menghitung vektor untuk perangkingan :

$$V_1 = \frac{1.7300}{1.7300 + 1.6313 + 1.2568} = 0,3746$$

$$V_2 = \frac{1.6313}{1.7300 + 1.6313 + 1.2568} = 0,3532$$

$$V_3 = \frac{1.2468}{1.7300 + 1.6313 + 1.2568} = 0{,}2721$$

Nilai terbesar adalah V_1 sehingga alternatif M_1 adalah alternatif yang terpilih sebagai alternatif terbaik.

2.5. Android

Android adalah sistem operasi pada *handphone* yang bersifat terbuka dan berbasis sistem operasi *linux*, *middleware*, dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri yang akan digunakan untuk bermacam perangkat bergerak. Awalnya, *Google Inc* membeli *Android Inc*, sebuah perusahaan pendatang baru yang membuat perangkat lunak untuk *ponsel*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance (OHA), konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi termasuk *Google*, HTC, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, dan *Nvidia* [4].

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkay seluler. Di lain pihak, Google merilis kode-kode Android di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler[4]. Berikut merupakan beberapa versi sistem operasi yang ada di android.

Karena pembangunan aplikasi perekomendasian konveksi ini menggunakan aplikasi bersifat mobile, dan setelah mengetahui hasil dari pertanyaan kuesioner yang disebarkan ke beberapa pelanggan. Ternyata penggunaan *smartphone mobile* android sangat banyak peminatnya, maka dari itu untuk pembangunan aplikasi yang tepat untuk digunakan oleh pelanggan ini bersifat *mobile* android, selain bisa mendapatkan aplikasinya gratis dan pada saat penggunaannya pun lebih cepat dengan hanya membuka aplikasinya saja pengguna bisa mengakses aplikasi tersebut tanpa harus mengetik terlebih dahulu alamat atau link yang akan dituju.

2.5.1.Android OS Version

1) *Android versi 1.1 (Tanpa Codename)*

   Pada tanggal 9 Februari 2009, Google merilis android versi 1.1 android versi ini dilengkapi dengan pembaharuan estetis pada aplikasi, jam

alarm, pencarian suara, pengiriman pesan dengan GMAIL, dan pemberitahuan email [4].

2) *Android versi 1. (Cupcake)*

Pada tanggal 27 April 2009, terdapat beberapa pembaharuan termasuk penambagan beberapa fitur dalam seluler versi ini, yakni kemampuan *multimedia*, dukungan *Bluetooth* A2DP, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem [4].

3) *Android versi 1.6 (Donut)*

Pada tanggal 15 September 2009, terdapat *update* yang menampilkan proses yang lebih baik dibandingkan sebelumnya, penggunaan baterai *indicator*, dan kontrol *applet* VPN [4].

4) *Android versi 2.0/ 2.1 (Éclair)*

Pada tanggal 26 Oktober 2009, *éclair* menjadi android pertama yang menghadirkan layanan navigasi Google Maps. Sistem tersebut menjadi awal mula era GPS yang sekarang bukan hanya ada di *ponsel*, tetapi di mobil-mobil modern. Eclaire juga menjadi Android pertama yang mendukung HTML5 pada peramban sehingga bisa memutar *video*. Kemampuan membuka layar atau *unlock-screen* dengan mekanisme menyapu atau *swipe* juga diperkenalkan pada *Éclair* [4].

5) *Android versi 2.2 (Froyo)*

Pada tanggal 20 Mei 2010, terdapat perubahan pada android versi 2.2, antara lain dukungan *adobe flash* 10.1, kecepatan kinerja dan aplikasi dua sampai lima kali lebih cepat, integrasi V8 *javascript* engine yang dipakai Google *Chrome* yang dapat mempercepat kemampuan *Rendering* pada *browser* [4].

6) *Android versi 2.3 (Gingerbread)*

Pada tanggal 6 Desember 2010, adanya peningkatan aplikasi dan penggunaan UI yang lebih baik [4].

7) *Android versi 3.0/ 3.1 (Honeycomb)*

Pada tanggal 15 Juli 2011, versi ini mendukung kemampuan tombol virtual untuk *home*, *back*, dan *menu*, untuk pertama kalinya. Sasarannya pun lebih ke peragkat tablet dibandingkan *smartphone* [4].

8) *Android versi 4.0 (Ice Cream Sandwich)*

Pada tanggal 19 Oktober 2011, versi ini mengambil kemampuan dari *very Honeycomb* tetapi lebih menyasar *smartphone*. Contohnya dalam kemampuan tombol virtual yang hingga saat ini banyak diimplementasikan para vendor. Beberapa pembaruan fitur lainnya mencangkup kemampuan membuka layar menggunakan wajah (*face unlock*), analisa penggunaan data internet, serta paket aplikasi bawaan dari vendor yang mencangkut kalender, mail, kalkulator, dan lainnya [4].

9) *Android versi 4.1, 4.2, 4.3 (Jelly Bean)*

Pada tanggal 27 Juni 2012, pada android versi ini *Jelly Bean* memungkinkan pengguna menggulir (*scroll*) cepat home screen ke bawah untuk melihat berbagai informasi penting, seperti agenda, email, dan laporan cuaca. Sebelumnya, pengguliran ke bawah hanya memperlihatkan *notifikasi* aplikasi. Selain itu versi ini merupakan upaya pertama Google untuk menghadirkan asisten digital yang dinamakan Google *Now*. Mulai dari versi ini, Google semakin berhasrat untuk membuat asisten digital yang lebih hidup, manusiawi, dan relevan bagi pengguna [4].

10) *Android versi 4.4 (Kitkat)*

Pada tanggal 31 Oktober 2013, android versi ini memberikan antarmuka berakses putih dan biru muda. Pada KitKat, Google menghadirkan perintah pencarian menggunakan suara atau disebut "*Ok*, *Google*". Fitur yang diberikan yaitu pesan singkat *Hangouts*, tetapi fitur *Hangouts* ini dilabeli sebagai layanan Google yang gagal karena tidak menuai penetrasi yang memuaskan [4].

11) Android versi 5.0 (Lollipop)

Pada tanggal 25 Juni 2014, desain yang diperhalus dan fitur-fitur yang diberikan sudah hadir pada versi-versi android sebelumnya lebih ditingkatkan. Pada versi ini mendukung untuk gambar berformat RAW, format ini memungkinkan para illustrator, fotografer, atau *graphic designer* menyimpan file dengan ukuran besar agar bisa diedit kembali tanpa mengurangi kualitas [4].

12) Android versi 6.0 (Marshmallow)

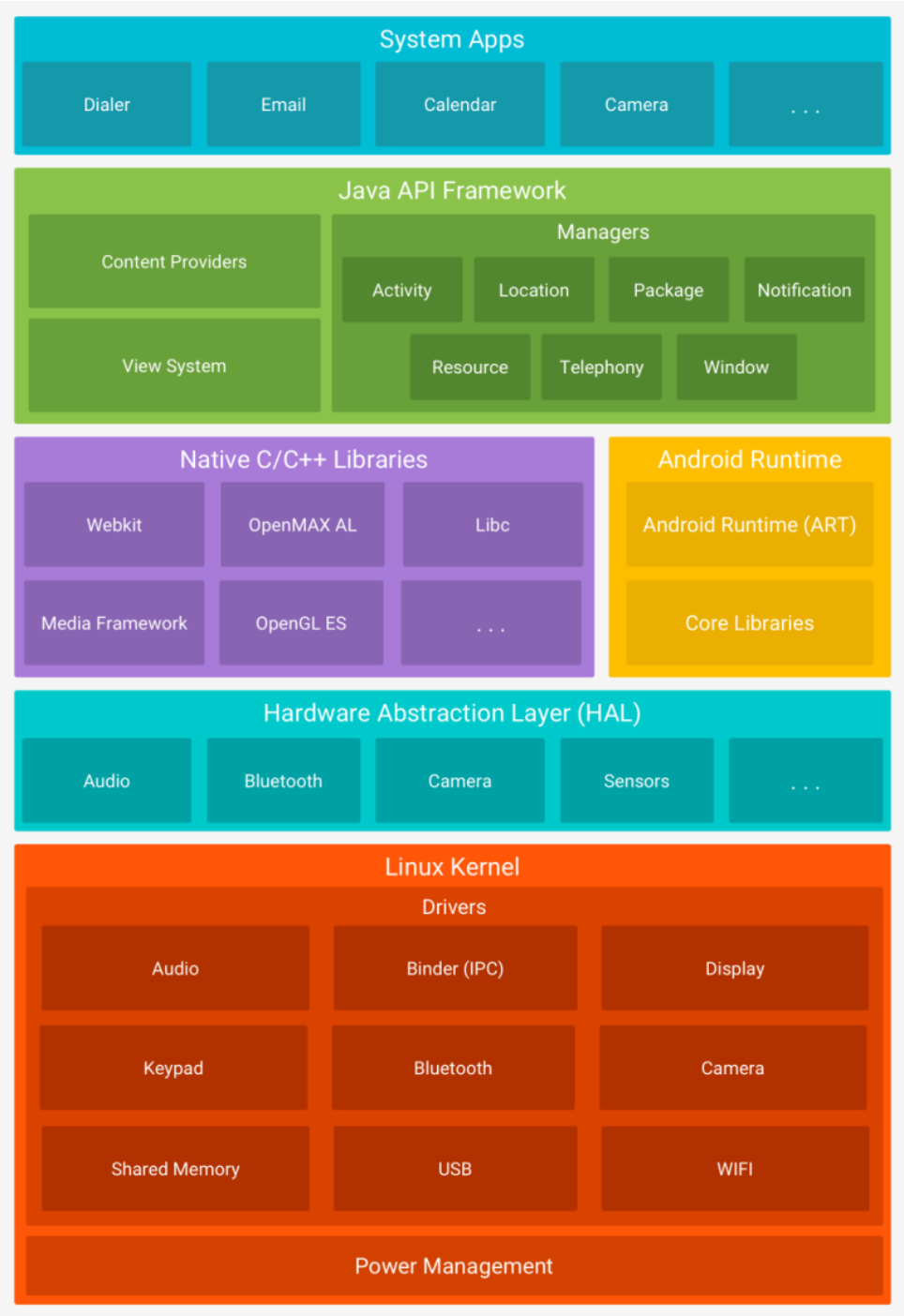
Pada tanggal 17 Agustus 2015, desain lebih dinamis dan fitur yang diberikan yaitu memory manager yang memungkinkan pengguna mengecek penggunaan memori pada setiap aplikasi dan rentan waktu pengecekannya bisa disetel dari tiga jam yang lalu hingga 24 jam sebelumnya [4].

13) Android versi 7.0, 7.1 (Nougat)

Pada tanggal 18 Juli 2016, adanya *Google Assistant* yang menggantikan *Google Now*. Asisten digital tersebut lebi bisa diandalkan untuk menjalankan berbagai fungsi. Fitur-fitur baru lainnya pada android versi ini mencangkup layar split-screen saat dipakai *multitasking*, serta fitur *Doze* yang telah dikenalkan di versi Android *Marshmallow* namun telah ditingkatkan. Android *Nougat* memiliki dukungan terhadap platform *virtual reality* terbaru Google [4].

2.5.2.Arsitektur Android

Android merupakan sistem operasi yang memiliki komponen arsitektur dengan diagram sebagai berikut :



Sumber gambar : https://developer.android.com

Gambar 2.1 Arsitektur Sistem Operasi Android

Berikut ini merupakan penjelasan dari arsitektur sistem android di atas :

1) Aplikasi

   Lapisan pertama dari arsitektur merupakan komponen yang digunakan oleh pengguna akhir ketika akan menjalankan program, pengguna hanya bisa melihat program tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dengan android *runtime* dengan menggunakan kelas dan *service* yang tersedia dari framework aplikasi. Lapisan aplikasi android berbeda dibandingkan dengan sistem operasi lainnya. Pada android semua aplikasi baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API *(Application Programming Interface)* yang sama.

2) Kerangka Kerja Aplikasi

   Kerangka kerja aplikasi ini menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu pada lapisan ini menyediakan abstraksi *generic* untuk mengakses perangkat, serta mengatur tampilan user *interface* dan sumber daya aplikasi.

3) Libraries

   Libraries merupakan lapisan paling dalam dari kerangka kerja android dan merupakan media *library* untuk memutar dan merekam berbagai macam format *audio* dan *video*, *Surface Manager* sebagai pengatur hak akses layer dari berbagai aplikasi, *Graphic Library* termasuk didalamnya SGL dan OpenGL sebagai tampilan 2D dan 3D, SQLite untuk mengatur relasi database yang digunakan pada aplikasi, SSI dan WebKit untuk browser dan keamanan internet.

4) Android Runtime

   Android runtime merupakan mesin virtual yang dirancang secara khusus untuk android ketika akan dijalankan pada lingkungan yang terbatas, dimana baterai yang terbatas, dimana baterai yang terbatas, CPU, memori, dan penyimpanan data.

5) Linux Kernel

Lapisan ini merupakan lapisan terakhir yang bertugas sebagai lapisan abstrak antara perangkat keras dan seluruh *software stack*. Dan merupakan bagian lapisan terdalam sebagai perantara akhir antar perangkat lunak dan perangkat keras.

Dari hasil penyebaran kuesioner ternyata banyak sekali pelanggan yang menggunakan smartphone android sebagai media komunikasi, maka dari itu untuk pembangunan aplikasi perekomendasian konveksi ini menggunakan berbasis android. Selain itu banyak vendor yang mengadaptasi android karena android ini bersifat *open source* atau terbuka pada sistem operasinya dan memudahkan dalam membuat aplikasi. Karena sumber kode perangkat lunaknya diberikan secara gratis tanpa harus membayar lisensi apapun.

2.6. Java

Java menurut definisi Sun merupakan sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer stand alone ataupun lingkungan jaringan. Bahasa pemrograman java ini bersifat *case sensitive*, sehingga harus lebih memperhatikan bentuk tertentu, sehingga dalam menuliskan semua baris *code* tersebut dalam satu baris tersendiri [5].

Pada bulan Agustus 1991, java dikembangkan dengan nama “Java 2” yang merupakan generasi kedua dari java platform. Pergantian nama baru tersebut diterapkan di setiap semua edisi Java diantaranya : *Standard Edition* (J2SE), *Enterprise Edition* (J2EE), dan *Micro Edition* (J2ME) [5]. Java memiliki tiga *platform* yang masing-masing mengarahkan sebagai tujuan tertentu dan sebagai lingkungan komputasi yang berbeda-beda :

1) *Standard Edition (J2SE)*

J2SE merupakan inti dari sebuah bahasa pemrograman Java. J2SE didesain sebagai jalan pada komputer desktop dan komputer *workstations*.

2) *Enterprise Edition (J2EE)*

J2EE berfungsi sebagai built-in dan mendukung untuk *servelts*. JSP dan XML, edisi ini bertujuan untuk aplikasi yang berbasis *server*.

3. *Micro Edition (J2ME)*

   J2ME didesain untuk piranti dengan keterbatasan memori, layar *display* terbatas dan power pemrosesannya terbatas juga.

Java dikenal sebagai bahasa pemrograman yang bersifat *strongly*, yang harus mendeklarasikan tipe data dari semua variable yang apabila salah dalam mengikuti aturan pendeklarasian variable, makan akan terjadi error pada saat proses kompilasi. Tipe data yang terdapat pada bahasa pemrograman Java pada umumnya yakni terdiri dari *integer, floating point*, char, dan *boolean* [5].

Dan untuk aturan penulisan pada bahasa pemrograman Java tidak jauh berbeda dengan bahasa pemrograman yang lain yakni harus diawali dengan huruf atau abjad, karakter mata uang, dan *underscore* (_) dan terdiri dari huruf atau abjad, angka,dan underscore (_). Dan tidak boleh menggunakan kata-kata yang dikenal oleh bahasa pemrograman Java (*keyword*), seperti *byte, case, int*, dan lainnya [5]. Penggunaan bahasa pemrograman Java pada pembangunan aplikasi perekomendasian konveksi yang dibuat ini, digunakan untuk membangun aplikasi untuk bagian *backend*.

2.7. Global Positioning System (GPS)

GPS *(Global Positioning System)* adalah sistem navigasi berbasis satelit yang bisa saling berhubungan yang berada diorbitnya. Satelit-satelit itu milik Departemen Pertahanan *(Departemen of Defense)* Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS reciver yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian dilayar pada peta elektronik [6].

Sejak tahun 1980, layanan GPS yang dulunya hanya untuk keperluan militer sekarang mulai terbuka untuk publik. Satelit ini mengorbit pada ketinggian sekitar 12.000 mil dari permukaan bumi. Posisi ini sangat ideal karena satelit dapat menjangkau area *coverage* yang lebih luas. Satelit ini akan selalu berada diposisi yang bisa menjangkau semua area di atas permukaan bumi sehingga dapat

meminimalkan terjadinya *blank spot* (area yang tidak terjangkau oleh satelit). Ada beberapa kategori GPS untuk pemakaiannya [6]:

1) *Lokasi*

   Digunakan untuk menentukan dimana lokasi suatu titik dipermukaan bumi berada.

2) *Navigasi*

   Membantu dalam mencari lokasi suatu titik di bumi.

3) *Tracking*

   Membantu memetakan posisi tertentu, dan perhitungan jaringan terdekat.

4) Timing

   Dapat dijadikan dasar penentuan jam seluruh dunia, karena memakai jam atom yang jauh lebih presesi dibanding dengan jam biasa.

GPS merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh sebuah alat penerima (*reciver*) di permukaan, dimana GPS *reciver* ini akan mengumpulkan informasi dari satelit GPS, seperti :

1. Waktu, GPS *reciver* menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
2. Lokasi, GPS memberikan informasi lokasi dalam tiga dimensi : *Latitude*, *Longitude*, Elevasi.
3. Kecepatan, ketika berpindah tempat GPS dapat menunjukan informasi kecepatan berpindah tersebut.
4. Arah perjalanan, GPS dapat menunjukan arah tujuan.
5. Simpan lokasi, tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS *reciver*.
6. Komulasi data, GPS *reciver* dapat menyimpan informasi *track*, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/ jam sampai tujuan, dan sebaginya.

Penggunaan GPS pada aplikasi perekomendasian konveksi yang akan dibuat ini digunakan sebagai tampilan peta dengan secara *detail*, terutama untuk memberikan penunjuk arah untuk mengetahui lokasi yang akan dituju. Dengan menerapkan teknologi GPS ini akan lebih memudahkan lagi bagi pelanggan.

2.8. IONIC

Ionic merupakan sebuah *framework* aplikasi mobile dengan berbasis HTML 5 dan dapat digunakan untuk mengembangkan aplikasi *mobile* dengan teknologi *web* seperti HTML, CSS, dan *JavaScript* yang ditargetkan untuk membangun aplikasi seluler hibrida. Dengan menggunakan *framework ionic* bisa membuat aplikasi lintas *platform* seperti Android. Ionic ini bersifat *open source* dan untuk tampilan antarmuka aplikasi, dimudahkan di *ionic* dengan bantuan ID-nya yang bernamaionic Lab. Karena *ionic* hanyalah sebuah *framework*, untuk merubah kodingan menjadi aplikasi Android perlu menggunakan *tools* tambahan.

Ionic adalah kumpulan elemen UI yang mencangkup banyak interaksi *mobile* dan merupakan komponen UI arahan dari *AngularJS*, dengan bersifat *open source* sebanyak mungkin, keduanya memiliki lisensi *open source* permisif yang dapat digunakan baik pada aplikasi komersil maupun *open souce*. Sebelumnya, pendiri *ionic* ingin membuat gebrakan revolusioner dengan konsep mengubah konsep *Write one Run Anywhere* dengan satu base *code* berbasis *javascript*. *Ionic* menggunakan *AngularJS* (*framework* MVC *full front-end*, 100% *Javascript*, 100% *client-side* dan kompatibel dengan desktop *browser* dan *mobile browser*), *Node*.js, SASS sebagai *engine*-nya.

Seperti *mobile framework* kebanyakan, *Ionic* juga memiliki komponen/elemen CSS standar digunakan untuk *smartphone*, seperti *button, list, card, form, range, tabs, grid* dan lainnya. Berikut ini merupakan kelebihan yang dimiliki *Ionic*:

1) *Ionic* Menggunakan lisensi *Open source*.
2) *Ionic platform* menggunakan lisensi *open source*, anda boleh membuat aplikasi *free* ataupun *commercial* dengan *Ionic*.
3) Menggunakan teknologi *web* terbaru.
4) *Ionic* memanfaatkan *AngularJS* untuk implementasi logicnya. Mengapa tidak memakai *jQuery*? karena *jQuery* terkenal lambat di *mobile*

sedangkan *Angular* menawarkan performa dan respon cepat serasa aplikasi *native*.

5) Target Hanya untuk Android 4 *danios* 7 Keatas
6) Anda bisa melihat ini sebagai kelebihan ataupun kekurangan. Bagi yang suka dengan hal baru dan ingin kode yang bersih tanpa memikirkan kompabilitas dengan versi *mobile* OS lama, maka *ionic* adalah *platform* yang paling pas.
7) Berbasis *Apache Cordova* /*Phonegap*.
8) *Ionic* hanya menyediakan *frameworknya*, untuk membungkusnya menjadi aplikasi Android atau iOS anda tetap pakai *phonegap*. Artinya bagi para programmer *phonegap* dengan *platform* lain, keahliannya tetap bisa dipakai.

2.9. JavaScript

Javascript diperkenalkan pada tahun 1995 yaitu suatu bahasa pemrogram untuk membangun suatu halaman website yang dapat bekerja di sebagian besar *browser* popular seperti *Internet Explorer*, *Mozzila Firefox*, *Netscape* dan *Opera*[12]. *Javascript* merupakan bahasa pemrograman yang diciptakan oleh *Brendan Eich* dan *Netscape* untuk menciptakan halaman web yang lebih interaktif dan dinamis. Diawal kelahirannya, *Javascript* lebih banyak digunakan di *web browser* dengan fungsionalitasnya seperti DOM *manipulation* untuk menciptakan web dinamis. Saat ini *Javascript* lahir di *browser Netscape Navigator* yang tidak lain menjadi pesaing dari *VBScript* yang dimiliki oleh *Internet Explorer*.

Javascript sudah memiliki lingkungan yang tidak hanya diperuntukkan oleh *web browser* saja, tetapi dapat digunakan untuk membuat aplikasi *desktop*, aplikasi *mobile*, dan aplikasi web sebagai *server side*. Didalam desktop terdapat *framework* seperti *TideSDK*, *AppJS*, dan IGNSDK yang dapat memfasilitasi pengembang *Javascript* untuk membuat aplikasi *desktop*, karena *Javascript* konvesional tidak dapat mengakses file sistem dan sistem operasi. Selain desktop, di ranah *mobile* pun bisa menggunakan *Javascript* yang digunakan sebagai pengembang aplikasi *mobile* seperti *Phonegap*, *Sencha*, dan *Titanium*. Bahkan ada suatu sistem operasi yang pengembang aplikasinya menggunakan *Javascript* dan HTML seperti *Firefox* OS.

Di server side dapat mengenal *NodeJS* yang dapat memfasilitasi pengembangan web untuk pengembangan web dalam membangun aplikasi dengan menggunakan baahasa pemrograman *Javascript*.

Pada pembangunan aplikasi ini menggunakan bahasa pemrograman *Javascript*, karena aplikasi yang dibangun bersifat *hybrid*, selain bahasanya yang dapat dipahami. *Javascript* juga dapat membuat sebuah aplikasi *mobile* yang bersifat *hybrid*.

2.10. Google Maps API

Google *Maps* API adalah sebuah library yang berasal layanan dari Google *Maps* yang berbentuk *JavaScript*, yang berisi fungsi-fungsi pemrograman yang bisa di integrasikan kedalam web atau aplikasi yang sedang dibuat[13]. Hanya dengan menggunakan *Key* Google *Maps* API, aplikasi yang dibuat dapat mengakses *maps* pada Google *Maps*.

Google *Maps* API dapat digunakan secara gratis dan tidak perlu untuk mengeluarkan biaya sebagai lisensi. Tetapi *request* maksimal pada sebuah peta hanya diperbolehkan sebanyak 2500 *request*/ hari. Jika lebih pengguna harus membeli lisensi Google *Maps* API untuk bisnis. Karena Google *Maps* API berbasis Web, maka untuk pembuatan aplikasinya hanya perlu menggunakan *tool Text Editor*, dan perangkat lain yang harus disiapkan yaitu *Browser*, dan koneksi internet.

Aplikasi perekomendasian konveksi ini pun membutuhkan sebuah tampilan peta sebagai media informasi sebuah lokasi yang akurat maka dari itu dengan menerapkan teknologi Google *Maps* API pada aplikasi yang akan dibangun, selain bisa menampilkan lokasi geografis pada peta Google. Pada teknologi Googel *Maps* API ini pun bisa mengidentifikasi keberadaan si pengguna aplikasi itu berada.

2.11. MongoDB

MongoDB adalah *database* NoSQL *(Not Only SQL)* VLDB *(Very Large Database) open source* yang menggunakan struktur data JSON untuk penyimpanan datanya. *MongoDB* dapat digunakan sebagai tempat penyimpanan data sistem tersebar (*distributed system*) karena kemampuan dalam menampung data yang

berjumlah sangat besar, *database* ini sering dipakai untuk aplikasi berbasis *Cloud*, *Grid Computing*, atau *Big Data*.

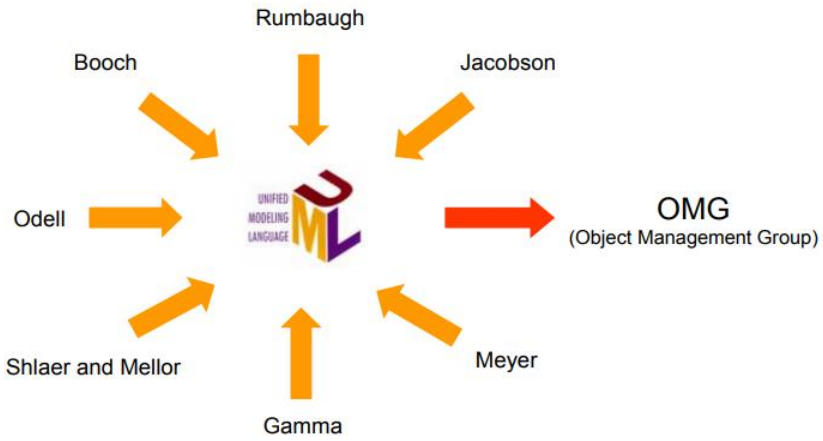
MongoDB didirikan oleh Kevin Ryan, Eliot Horowitz, dan Dwight Merima, mereka bergabung di *MongoDB Inc*. *MongoDB* memiliki fitur *high performance*, *high availability*, dan *automatic scaling*. *MongoDB* menggunakan bahasa pemrograman *JavaScript* sebagai operasi CRUD *(Create, Read, Update, Delete)* , agregasi, *indexing*, dan operasi *database* lainnya. *MongoDB* tidak menggunakan *table*, tetapi menggunakan dokumen sebagai media penyimpanan datanya yang berstruktur seperti JSON.

Pada aplikasi perekomendasian konveksi ini menggunakan *MongoDB* sebagai databasenya, karena lebih cepat dalam performanya pada saat *backup data* secara *realtime*, bisa memecah database yang besar menjadi beberapa bagian demi optimalisasi performa databasenya dan berguna ketika aplikasi yang dibuat ini membutuhkan sebuah database kapasitasnya lebih besar.

2.12. Unified Modelling Language (UML)

UML (Unified Modelling Language) adalah bahasa pemodelan grafis yang digunakan untuk menspesifikasi, visualisasi, desain dan mendokumentasikan sistem perangkat lunak yang akan dibangun, yang pembangunannya menggunakan pemrograman yang berorientasi objek. UML memenuhi syarat dalam objek analisis dan desain karena termasuk kedalam diagram alternatif untuk menjelaskan statis properti, pengguna sistem atau komponen, dan sistem arsitektur [7].

Sejarah UML cukup panjang, pada era tahun 1990 beragamnya metodologi pemodelan berorientasi objek ini bermunculan, diantaranya : metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dan lain sebagainya. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi tersendiri, yang mengakibatkan timbul masalah baru apabila akan bekerja sama dengan group atau perusahaan lain yang menggunakan metodologi yang berlainan.



Sumber gambar : https://ilmukomputer.com

Gambar 2.2 Macam-macam metodologi pada UML

Dimulai pada bulan Oktober 1994 Booc, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang metodologinya sering digunakan mempelopori usaha penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 diterbitkan draft pertama dari UML (versi 0.8). Sejak taun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group (OMG – http://www.omg.org)*.

Pada tahun 1997 UML versi 1.1 muncul, dan saat ini versi tersebut adalah versi 1.5 yang dirilis bulan 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999 [7]. Sejak itulah UML telah menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. Berikut merupakan tujuan dan fungsi dari penggunaan UML :

1) Dapat memberikan bahasa visual kepada pengguna dengan berbagai macam pemrograman maupun proses rekayasa.
2) Dapat berguna sebagai blueprint, karena sangat lengkap dan detail dalam menggambarkan sistem yang akan dibuat terutama untuk mengetahui alur sistem mulai dari dari awal hingga akhir, implementasi, dan perangkat apa saja yang dibutuhkan untuk pembangunannya.

Karena aplikasi yang akan dibangun ini beorientasi obyek maka untuk perancangan konsep pembangunan aplikasi ini menggunakan pendekatan sistem *Object Oriented* yang divisualisasikan dengan UML. Didalam UML memiliki beberapa

jenis diagam diantaranya adalah sebagai berikut : *Use Case Diagram, Activity Diagram, Class Diagram, Sequence Diagram* [7].

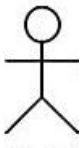
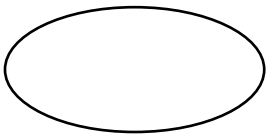
2.12.2. Use Case Diagram

Use case diagram merupakan gambaran fungsionalitas dari beberapa atau semua *actor*, *use case*, dan interaksi untuk memperkenalkan suatu sistem. *Use case* diagram tidak menjelaskan secara detail tentang penggunaan *use case*, tetapi hanya memberikan gambaran singkat tentang hubungan *use case*, *actor*, dan sistem. Didalam *use case* hanya diketahui fungsi-fungsi yang berkaitan dengan pembangunan sistem yang akan dibuat.

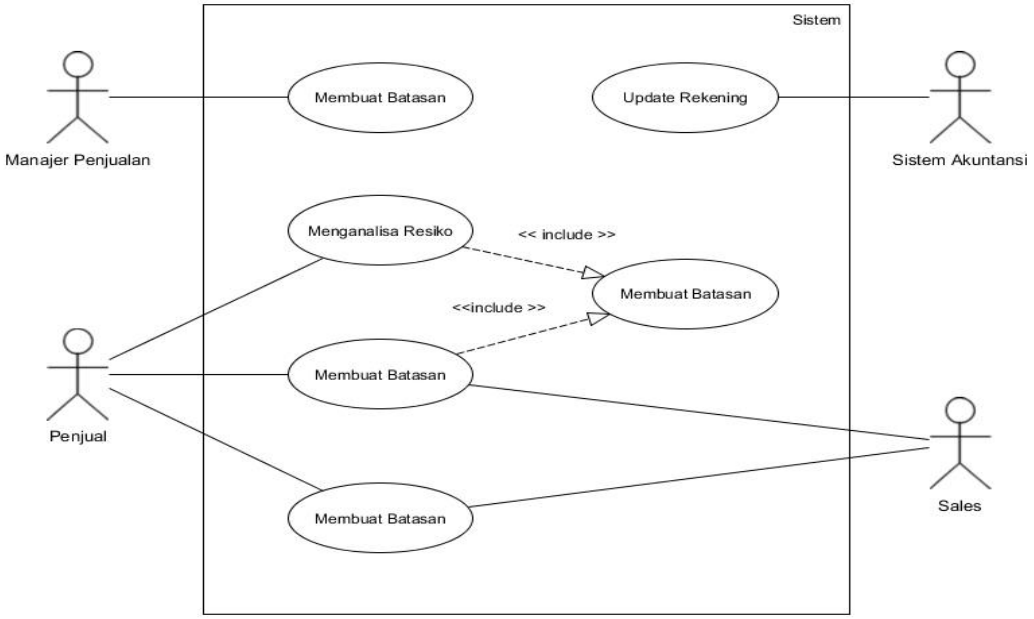
Sebuah *use case* dapat di *include* oleh lebih dari suatu *use case* lain, shingga duplikasi fungsionalitas dapat dihindari dengan menarik keluar fungsionalitas yang *common*. Use case juga dapat meng *extend use case* lain dengan behaviornya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa *use case* yang satu merupakan spesialis dari yang lain. Diagram *use case* memiliki beberapa *element* yang digunakan untuk menggambarkan rancangan dan penghubung antar komponen, baik *actor* dengan *use case* atau *use case* dengan *use case* yang lainnya. Untuk lebih jelasnya bisa dilihat pada Table 2.1 beserta contoh *use case* diagram pada Gambar 2.2 :

Tabel 2.1 Notasi *Use Case Diagram*

Bentuk Notasi	Nama	Keterangan
	*Asosiasi*	Hubungan dan ketergantungan setiap aktor dengan *use case* yang dijalani
--- <include> --→	*Include*	Digunakan untuk menambahkan use case dengan use case yang lain. Opsional
--- <extend> --→	*Extend*	Digunakan untuk menyederhanakan satu use case yang digunakan dan harus dijalankan pada use yang lain.

	*Actor*	Berfungsi untuk mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Actor hanya berinteraksi dengan use case tetapi tidak memiliki kontrol atas use case.
	*Use Case*	Gambaran fungsionalitas dari suatu sistem

Berikut contoh perancangan diagram *Use Case* :



Gambar 2.3 Contoh Diagram Use Case

2.12.3. Activity Diagram

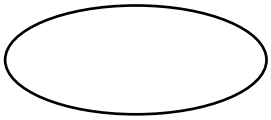
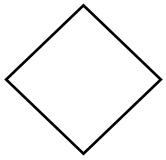
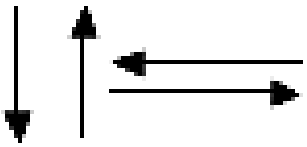
Activity Diagram atau Diagram Aktivitas digunakan untuk mendokumentasikan alur kerja pada sebuah sistem yang akan dibangun, yang dimulai dari pandangan *business level* hingga ke operasional *level*, adapun penjelasannya antara lain :

1) *Activity diagram* merupakan cara lain untuk memodelkan aliran kejadian.

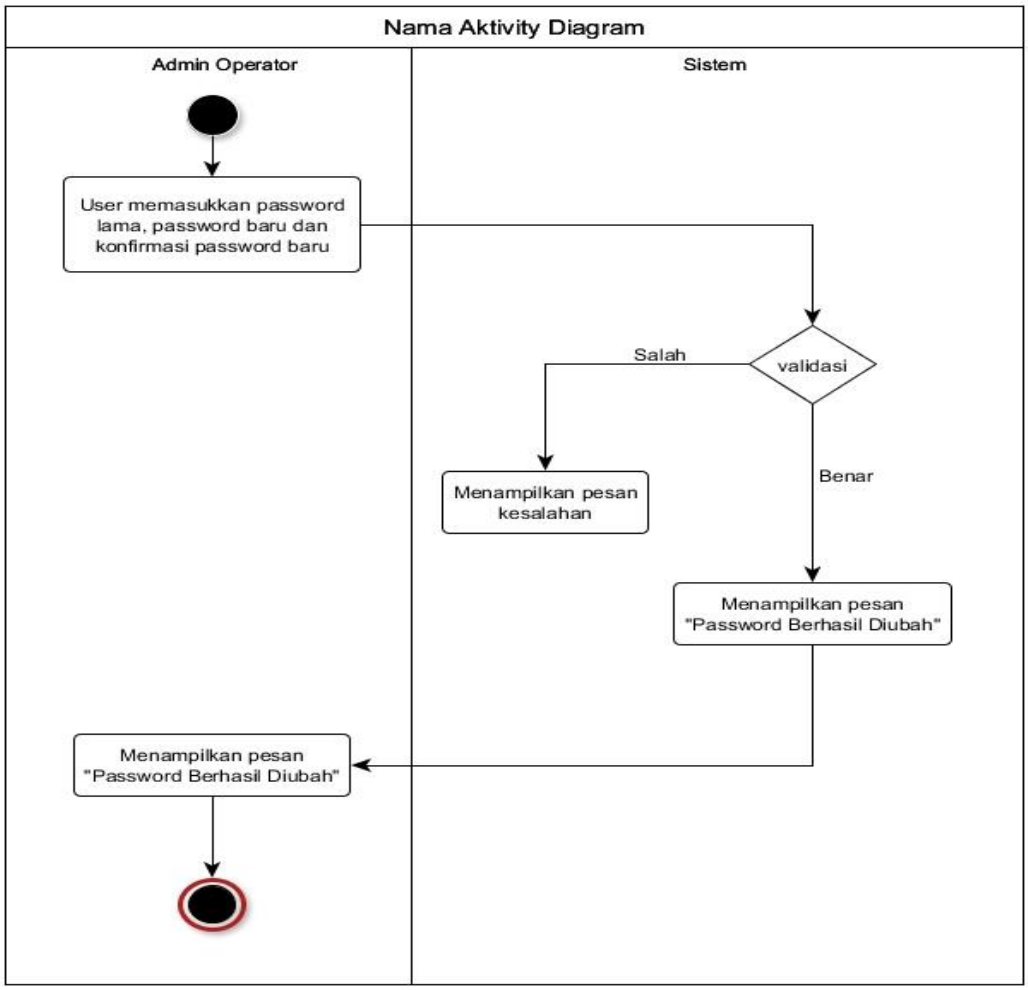
2) Berfungsi untuk menggambarkan *workflow* atau aliran kerja proses bisnis.
3) *Activity diagram* menggambarkan berbagai aliran aktivitas dalam sebuah sistem yang sedang dirancang bagaimana masing aliran berawal, decision yang mungkin terjadi, dan bagaimana alur berakhir.
4) *Activity diagram* dapat menggambarkan proses *parallel* yang mungkin dapat terjadi pada beberapa eksekusi.

Berikut merupakan tabel penjelasan notasi *activity diagram* :

Tabel 2.2 Notasi *Activity Diagram*

Bentuk Notasi	Nama	Keterangan
	*Initial Node*	Memperlihatkan bagaimana masing-masing kelas antar muka saling berinteraksi satu sama lain.
	*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	*Activity*	Bagaimana objek dibentuk atau diawali.
	*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri.
	*Decision*	Digunakan untuk menggambarkan suatu keputusan atau tindakan yang harus diambil pada kondisi tertentu.
	*Line Connector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

Berikut merupakan contoh gambaran dari *activity diagram* :



Gambar 2.4 Contoh *Activity Diagram*

2.12.4. Class Diagram

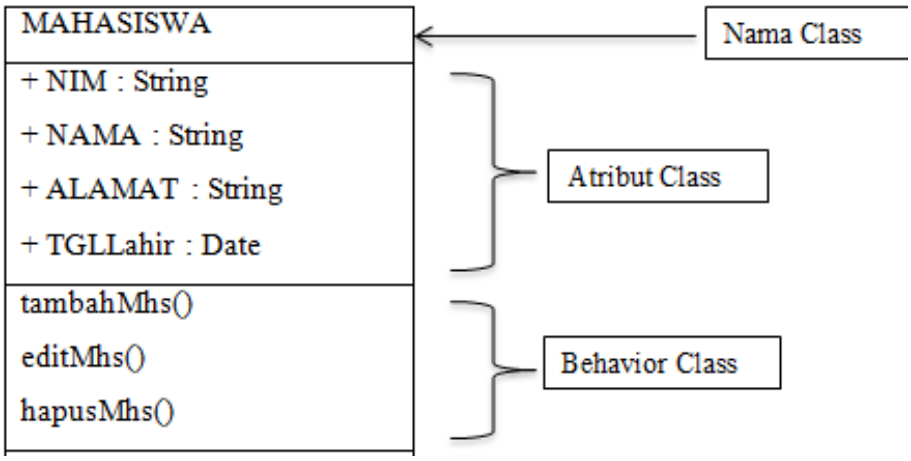
Class Diagram merupakan elemen terpenting dalam sistem berorientasi objek, kelas mendeskripsikan satu blok pembangun sistem. *Class Diagram* memiliki fitur yang memodelkan multiplisitas, ketampakan, penanda, *polymorphism*, dan karakter-karakter lainnya. UML menggunakan istilah fitur sebagai istilah umum yang meliputi *property* dan operasi sebuah kelas Terdapat empat bagian masing-masing bagan kelas, yaitu : nama, atribut, perilaku, dan keterangan kelas. Berikut merupakan tabel penjelasan notasi *activity diagram* :

Tabel 2.3 Notasi *Class Diagram*

Bentuk Notasi	Nama	Deskripsi
Nama *Class* + atribut + atribut + atribut + method + method	*Class*	*Class* adalah blok-blok pembangunan pada pemrograman berorientasi obyek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari suatu *class*. Bagian tengah mendefinisikan method-method dari sebuah class.
———	*Association*	Sebuah asosiasi merupakan sebuah relationship paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe relationship dan juga dapat menampilkan hokum-hukum multiplisitas pada sebuah relationship.

	*Composition*	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.
	*Dependency*	*Dependency* digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Sebuah dependency dilambangkan sebagai sebuah panah bertitik-titik.
	*Aggregation*	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi.

Gambar berikut merupakan contoh perancangan *class diagram* kelas :

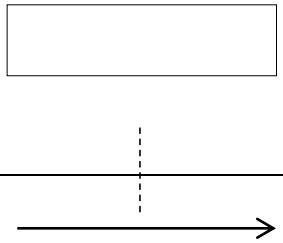


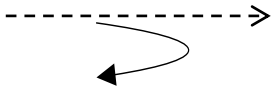
Gambar 2.5 Contoh *Class Diagram*

2.12.5. Sequence Diagram

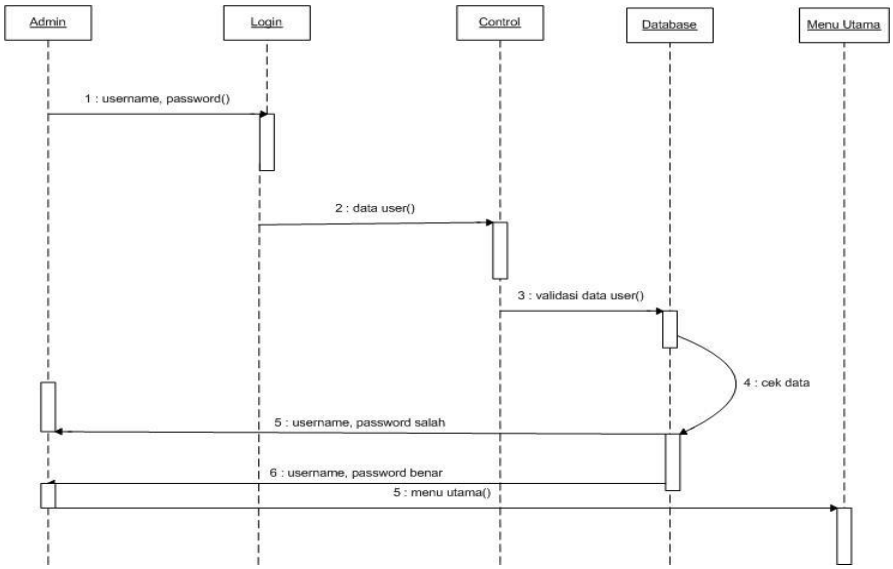
Diagram Sequence digunakan untuk memberikan gambaran interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. Sequence diagram terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Dan *sequence diagram* ini bisa digunakan untuk membuat rangkaian langkah-langkah yang dilakukan sebagai respon dari *event* menjadi keluaran (*output*) tertentu. Dengan diawali dengan men-trigger aktivitas, proses dan perubahan yang terjadi secara internal dan output apa yang dihasilkan. Setiap objek termasuk aktor memiliki lifeline vertikal. Pesan digambarkan sebagai garis panah dari satu objek ke objek yang lainnya. Berikut merupakan komponen-komponen yang digunakan untuk perancangan *sequence diagram* :

Gambar 2.4 Notasi *Sequence Diagram*

Bentuk Notasi	Nama	Keterangan
	Object	Menambahkan objek baru pada diagram
	Object message	Menggambar pesan (message) antar dua objek

	Message to selft	Menggambar pesan *(message)* yang menuju dirinya sendiri
	Return message	Menggambarkan pengembalian dari pemanggilan prosedur
	Time Active	Menyatakan objek dalam keadaan aktif dan berinteraksi dengan pesan

Contoh perancangan *sequence diagram* :



Gambar 2.6 Contoh *Sequence Diagram*

2.13. Metode Pengujian

Metode pengujian adalah suatu teknik menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan. Perangkat lunak dapat diuji dengan dua cara, yaitu :

2.13.1. Pengujian *Whitebox*

Pengujian *White Box* menurut Roger. S. Pressman *"White Box Testing is a test case design method that used the control structure of te*

procedural design to derive test case". Teknik pengujian ini merupakan suatu Metode Basis *Path* yang digunakan untuk menentukan ukuran kompleksitas logika dari suatu logika. Metode *White Box* yang digunakan berguna sebagai :

1) Mengukur kompleksitas *logic* dari desain prosedur dan sekaligus sebagai pedoman untuk mendapatkan konsisten jalur aplikasi.
2) Pengujian yang dilakukan dijamin menggunakan *statement* dalam program minimal satu kali selama pengujian.
3) Menghitung *cyclometris complexity* sebagai ukuran kontitif untuk menentukan jumlah *independent* path sebagai jalur yang perlu diuji.
4) Menguji struktur data internal.

2.13.2. Pengujian *Blackbox*

Pengujian *Black Box* merupakan pendekatan komplementer dari teknik *white box*, pengujian *black box* ini diharapkan mampu mengungkapkan kesalahan yang lebih luas dibandingkan dengan teknik pengujian *white box*. Pengujian *black box* berfokus pada pengujian persyaratan fungsional dari sistem, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Uji coba *black box* bukan merupakan alternative dari uji coba *white box*, tetapi merupakan suatu pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *white box*. Pada pengujian ini 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.

Berbeda halnya dengan metode *white box* yang dilaksanakannya diawal proses, uji coba *black box* diaplikasikan dibeberapa tahapan berikutnya. Karena uji coba *black box* ini mengabaikan struktur kontrol, sehingga perintahnya hanya difokuskan pada informasi domain.

Dengan menggunakan kedua metode pengujian tersebut, dapat membantu aplikasi perekomendasian ini ini mengetahui suatu kesalahan atau kekurangan dari sebuah aplikasi, dan dengan begitu bisa diperbaiki kembali agar aplikasi yang dibuat ini bisa sesuai dengan target yang diinginkan.