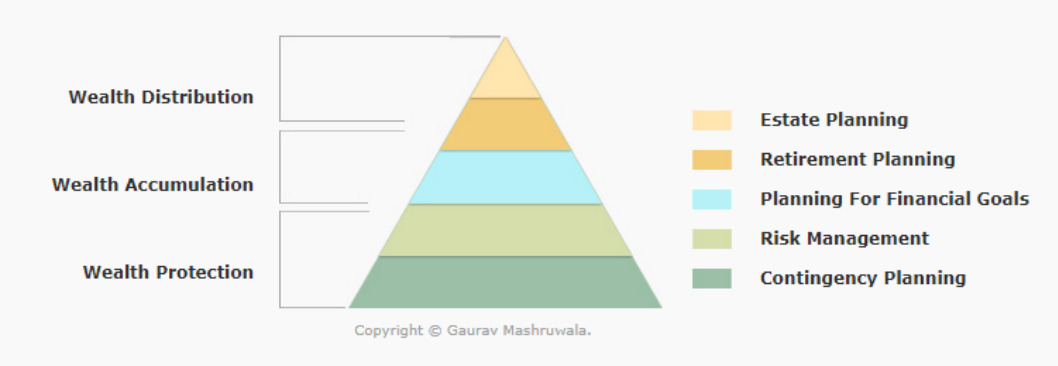
BAB 2
LANDASAN TEORI

2.1 Landasan Teori

Landasan teori merupakan definisi dan gambaran dari konsep yang telah disusun secara sistematis dari penelitian ini dan menjadikannya dasar yang kuat. Landasan teori yang digunakan dalam penyusunan Pembangunan Aplikasi Perencanaan Keuangan Pribadi menggunakan *teknologi Firebase Cloud Messaging* dan API Toko Online berbasis android ini meliputi perencanaan keuangan, metode merencanakan keuangan, aplikasi *mobile*, android, kelebihan dan kekurangan android, arsitektur android, *Firebase Cloud Messaging*, Toko Online API, Android Studio, *Unified Modeling Language* (UML).

2.2 Perencanaan Keuangan

Perencanaan merupakan proses bagaimana mencapai tujuan dengan memperhatikan batasan – batasan yang dimiliki maupun yang tidak boleh dilakukan demi mencapai suatu tujuan. Perencanaan keuangan adalah proses untuk mencapai tujuan keuangan melalui manajemen keuangan yang terencana dengan baik sehingga apa yang direncanakan dapat diwujudkan. Tentunya dalam merencanakan keuangan dapat diurutkan berdasarkan prioritasnya dan digambarkan melalui piramida seperti pada gambar 2.1 dibawah ini.



Gambar 2.1 Piramida Prioritas Perencanaan Keuangan

(Sumber : https://www.gauravmashruwala.com/financial-planning/10-April-2019)

Berikut merupakan penjelasan dari gambar piramida prioritas keuangan diatas.

1. *Contingency Planning*

   Prioritas pertama dalam perencanaan keuangan adalah memenuhi kebutuhan jangka pendek, dalam hal ini adalah mengelola arus kas (*Cash Flow)*, dana darurat, dan dana hutang yang harus dipastikan sehat
2. *Risk Management*

   Prioritas kedua adalah manajemen risiko individu (*personal risk management*). Pada bagian ini, seseorang memastikan aset-asetnya telah terlindungi dengan cukup, dengan kata lain seseorang memastikan dirinya terlindungi oleh asuransi atau proteksi.
3. *Planning For Financial Goals*

   Prioritas ketiga adalah tujuan – tujuan keuangan jangka menengah. Prioritas yang ada bisa saja berbeda – beda tergantung dari keinginan yang ingin dicapai seperti merencanakan membeli rumah , mobil dan jalan – jalan.
4. *Retirement Planning*

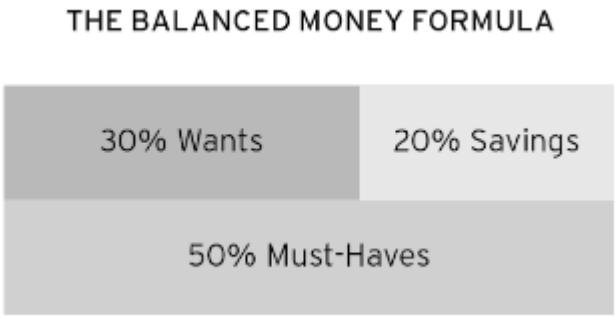
   Prioritas keempat merupakan tujuan keuangan jangka panjang, membuat dana prioritas seperti dana hari tua
5. *Estate Planning*

   Akhir dari tahap piramida prioritas keuangan yaitu prioritas terakhir waris atau distribusi keuangan.

2.3 Metode 50/30/20

Merencanakan keuangan sebenarnya mudah untuk dilakukan jika mengerti akan kemampuan dan batasan. Kemampuan yang dimaksud yaitu mampu untuk membeli sesuatu selama barang tersebut masih terjangkau oleh kondisi uang yang dimiliki. Serta batasan yang dimaksud yaitu membeli barang barang sesuai dengan kebutuhan dan tidak membeli barang – barang secara berlebihan. Merencanakan keuangan juga membutuhkan langkah langkah yang tepat dalam mengatur porsi – porsi uang yang akan digunakan dan yang akan ditabung demi tercapainya tujuan.

Merencanakan keuangan juga perlu metode yang tepat agar terhindar dari salah perencanaan yang berimbas pada kondisi keuangan serta tujuan yang dicapai. Pakar kebangkrutan dari Universitas Harvard Elizabeth Warren menciptakan metode 50/30/20 dalam mengatur keseimbangan keuangan. Metode 50/30/20 merupakan metode yang sederhana, tepat sasaran dan efektif.



Gambar 2.2 Metode Mengatur keuangan 50/30/20

(Sumber : All Your Worth : The Ultimate Lifetime Money Plan [11])

Metode 50/30/20 merupakan cara untuk mengatur keuangan supaya perencanaan yang dilakukan berhasil. Metode ini menggunakan rasio pos keuangan yang dipisahkan dengan menggunakan tiga pos pengeluaran yaitu kebutuhan dasar, tabungan atau investasi dan hiburan. Berikut adalah penjelasan dari metode 50/30/20.

1. 50% untuk kebutuhan dasar atau kebutuhan sehari – hari

   Uang yang didapat digunakan 50% untuk kebutuhan sehari – hari seperti biaya sewa tempat tinggal, biaya cicilan berbagai macam tagihan, biaya

untuk makan, biaya untuk belanja keperluan bulanan seperti alat mandi, biaya asuransi, biaya untuk penunjang elektronik seperti pulsa dan internet, biaya listrik.

2. 30% untuk hiburan

   Setelah tercukupi kebutuhan dasar dengan anggaran 50% dari pendapatan tentunya perlu juga akan kebutuhan hiburan. Biaya untuk hiburan sebesar 30% seperti kebutuhan akan belanja, jalan – jalan, nonton dan sebagainya. Dengan diporsirnya kebutuhan akan hiburan ini dapat terkendali biaya yang dipakai untuk hiburan berapa besar yang harus dikeluarkan sehingga tidak merasa uang yang didapat habis begitu saja tanpa tahu digunakan untuk apa.

3. 20% untuk tabungan atau investasi

   Pos ini ditunjukan untuk memenuhi berbagai tujuan keuangan yang ingin diperoleh atau yang ingin digunakan di kemudian hari, dengan porsi 20% untuk biaya tabungan atau investasi misalnya untuk memenuhi kebutuhan dana darurat ataupun asuransi, menabung untuk membeli sesuatu atau berinvestasi.

Rumus keuangan yang seimbang merupakan perencanaan yang tepat dalam mengatur tujuan rencana keuangan [11]. Berikut penjelasan mengapa harus dengan formula 50/30/20 :

1. 50% untuk kebutuhan dasar atau untuk kebutuhan sehari –hari karena kebutuhan hidup itu terus berlanjut, setiap orang memiliki kebutuhannya masing – masing dan itu akan terus berkelanjutan. Merasa aman karena kebutuhan dasar sudah dipenuhi. Dan alasan terakhir adalah teruji dengan berjalannya waktu.
2. 30% untuk kebutuhan hiburan, karena setiap orang pantas untuk mendapatkan waktu untuk hiburannya masing – masing akan uang yang didapatnya dan kehidupan tidak hanya untuk bekerja keras saja. selama menghabiskan tidak lebih dari 30% untuk kebutuhan hiburan maka kondisi keuangan dapat dikontrol.

3. 20% untuk ditabung atau investasi, karena setiap orang memiliki tujuan dalam rencana keuangan yang dilakukan, oleh karena itu menabung sedikit demi sedikit lama – lama akan menjadi banyak dan tidak terasa jika telah menabung.

Tentunya metode 50/30/20 ini akan efektif jika diterapkan terus menerus. metode ini juga dapat diterapkan untuk segala kalangan selama kebutuhan pengeluarannya tidak lebih besar dari pada pendapatan. dengan berhemat merupakan cara yang terbaik untuk mencapai tujuan keuangan yang diinginkan. Metode ini pula yang akan digunakan dalam penelitian skripsi ini karena cukup baik dan jelas dalam merancang dan merencanakan keuangannya.

2.4 Aplikasi Mobile

Aplikasi mobile berasal dari kata *Application* yang berarti suatu bagian dari perangkat lunak komputer yang dibuat untuk mendukung suatu pekerjaan atau aktivitas yang dilakukan manusia. Serta *Mobile* dapat diartikan sebagai suatu perpindahan dari satu tempat ke tempat lain. Aplikasi mobile dapat diartikan sebagai perangkat lunak yang berjalan di handphone atau tablet PC. Dengan menggunakan aplikasi mobile dapat mempermudah berbagai macam aktivitas [12].Perangkat mobile memiliki banyak jenis dalam ukuran desain dan tampilan namun memiliki karakteristik yang berbeda dengan *desktop system.* Selain itu perangkat mobile juga memiliki ukuran yang kecil serta memori yang terbatas.

Dengan perangkat mobile pengguna dapat melakukan berbagai macam aktivitas seperti hiburan, maupun pekerjaan. Pemanfaatan aplikasi mobile untuk hiburan paling banyak digemari oleh hampir 70% pengguna telepon seluler, karena dengan memanfaatkan adanya fitur *game*, *music player*, sampai video *player* membuat kita menjadi semakin mudah menikmati hiburan kapan saja dan dimanapun [12].

2.5 Android

Android adalah sistem operasi berbasis linux yang dirancang untuk perangkat bergerak layar sentuh seperti *smartphone* dan komputer tablet. Android merupakan sebuah kumpulan perangkat lunak untuk perangkat mobile yang mencangkup sistem operasi, middleware dan aplikasi utama mobile [13]. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasinya sendiri untuk digunakan oleh berbagai macam piranti bergerak [14]. Android awalnya dikembangkan oleh Android, Inc,. dan kemudian dibeli oleh Google pada tahun 2005. Lalu secara resmi di rilis pada tahun 2007 bersamaan dengan di dirikannya Open Handset Alliance, konsorsium dari perusahaan – perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Dan memulai penjualannya pertamanya pada tahun 2008.

Android memiliki empat karakteristik [13]. Diantaranya :

1. Terbuka

   Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera dan lain-lain. Android merupakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. Android merupakan *open source*, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. *Platform* ini akan terus berkembang untuk membangun aplikasi *mobile* yang inovatif.
2. Semua Aplikasi Dibuat Sama

   Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (*third-party application*). Semua apliksi dapat dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.
3. Memecahkan Hambatan Pada Aplikasi

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dpat menggabungkan informasi yang diperoleh dari *web* dengan data pada ponsel seseorang seperti kontak pengguna, kalender atau lokasi geografis.

4. Pengembangan Aplikasi Yang Cepat dan Mudah

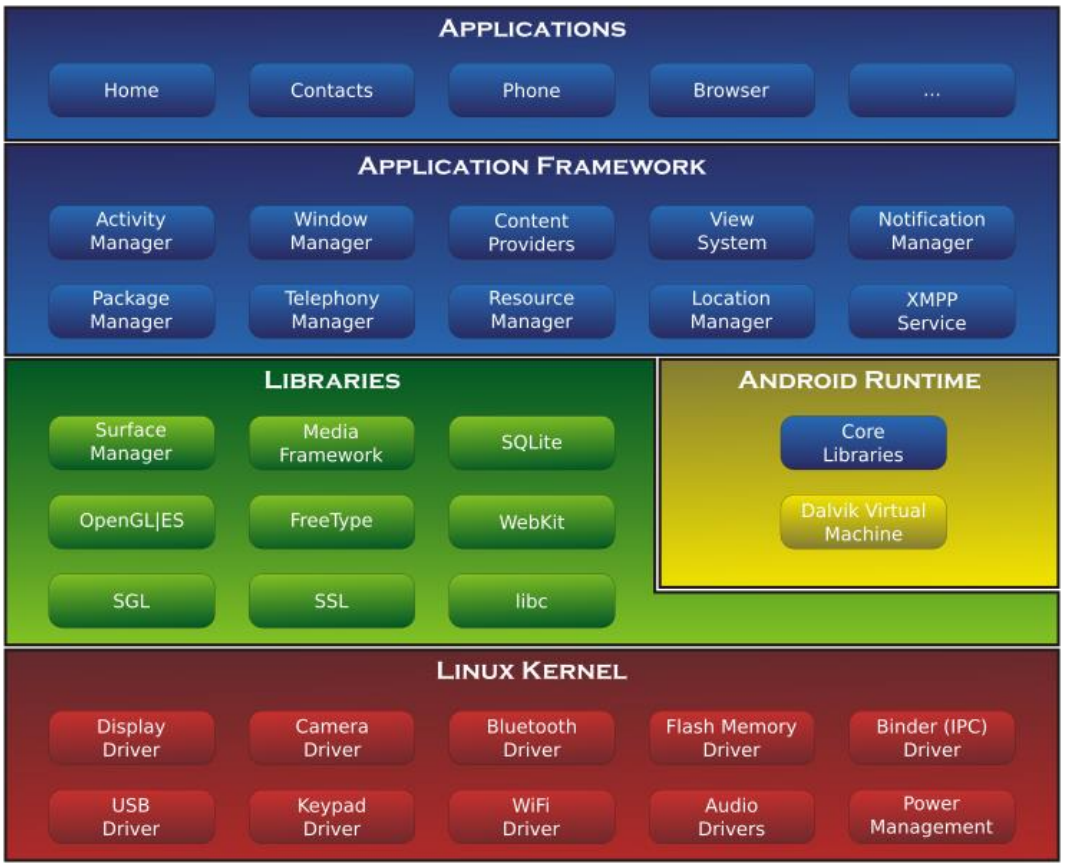
   Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan aplikasi yang semakin baik. Android memiliki sekumpulan *tools* yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat.

Google Inc. sepenuhnya membangun android dan menjadikannya open source sehingga para pengembang dapat menggunakan android tanpa biaya untuk lisensi dari google dan dapat membangun tanpa adanya batasan - batasan untuk membangun aplikasi yang semakin baik. Android *Software Development Kit* (SDK) menyediakan alat dan *Application Programming Interface* (API) yang diperlukan untuk mulai mengembangkan apliaksi pada platform Android menggunakan bahasa pemrograman java [15]. Adapun versi android yang sudah dirilis :

a. Android versi 1.1
b. Android versi 1.5 (cupcake)
c. Android versi 1.6 (Donut)
d. Android versi 2.0 - 2.1 (Éclair)
e. Android versi 2.2.3 (Froyo)
f. Android versi 2.3 – 2.3.7 (Gingerbread)
g. Android versi 3.0 – 3.2.6 (Honeycomb)
h. Android versi 4.0 – 4.0.4 (Ice Cream Sandwich)
i. Android versi 4.1 – 4.3.1 (Jelly Bean)
j. Android versi 4.4 (Kitkat)
k. Android versi 5.0 (Lollipop)
l. Android versi 6.0 (Marshmallow)
m. Android versi 7.0 (Nougat)
n. Android versi 8.0 (Oreo)

2.6 Arsitektur Android

Arsitektur android terdiri dari berbagai lapisan dan setiap lapisan terdiri dari beberapa program memiliki fungsi yang berbeda. Seperti gambar 2.3 dibawah ini :



Gambar 2.3 Arsitektur Android

(Sumber : https://id.wikipedia.org/wiki/Android_(sistem_operasi)/10-April-2019)

Berikut adalah penjelasan dari gambar diatas :

1. *Applications*

   Applications ini adalah layer dimana berhubungan dengan aplikasi saja, Aplikasi berada pada lapisan terluar dari Arsitektur Android. Lapisan ini berjalan dalam *Android runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi[16]. Pengguna berinteraksi dengan lapisan ini untuk fungsi umum seperti menelepon, mengakses website.
2. *Application Frameworks*

Application Framework yaitu kerangka aplikasi yang menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Pada layer ini biasanya digunakan oleh *developer* atau programmer Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi. Komponen – komponen yang termasuk ke dalam *Application Frameworks* yaitu :

a. *View System* digunakan untuk membuat *user interface* aplikasi
b. *Activity Manager* mengontrol semua aspek dari siklus hidup aplikasi dan *activity stack*
c. *Content Providers* mengizinkan aplikasi untuk mempublikasikan dan berbagi data dengan aplikasi lainnya.
d. *Resource Manager* memberikan akses kepada *resources* yang bukan kode seperti *string*, setting warna, dan *layout user interface*
e. *Notification Manager* membuat aplikasi dapat menampilkan pengingat dan notifikasi kepada pengguna.

3. Libraries

Libraries ini adalah *layer* dimana fitur – fitur android berada. *Developer* mengakses libraries untuk menjalankan aplikasinya. Libraries membawa sekumpulan instruksi untuk mengarahkan perangkat android dalam menangani berbagai tipe data. berikut adalah beberapa kegunaan *libraries* :

a. *Surface Manager libraries* untuk mengolah tampilan layar windows
b. SGL dan OpenGL *libraries* grafik untuk 2D dan 3D
c. Media *Framework Libraries* untuk media pemutaran media video dan audio
d. SQL *Lite* untuk Database
e. SSL dan *Webkit* untuk integrasi dengan *web browser* dan *security*
f. *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's.
g. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embedded webview*

4. Linux Kernel

Linux kernel merupakan layer inti dari semua sistem android berada. Karena dilapisan inilah yang memberikan fungsi – fungsi *Driver, Memory management, security* pada sistem android.

2.7 Firebase

Firebase adalah *Cloud Service Provider* dan *Backend as a Service* yang dimiliki oleh google. *Firebase* merupakan solusi yang ditawarkan oleh Google untuk mempermudah dalam pengembangan aplikasi mobile maupun web. *Firebase* memiliki banyak SDK yang memungkinkan untuk mengintegrasikan layanan dengan Android, Ios, Javascript, C++, hingga Unity. Berikut fitur dalam firebase pada gambar 2.4 dibawah ini.

Gambar 2.4 Fitur-fitur dalam *Firebase*

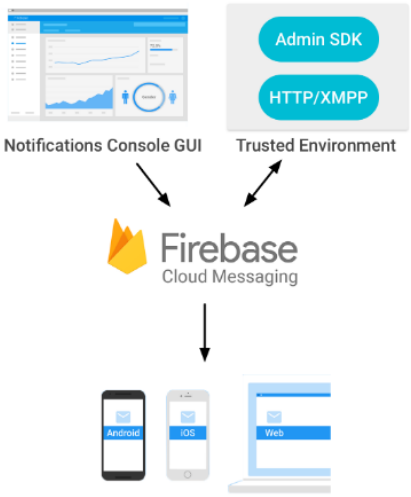
(Sumber : https:// firebase.google.com/10-April-2019)

Gambar diatas merupakan fitur-fitur yang ada didalam *firebase*. terdapat fitur untuk *Develop, Grow,* dan *Earn.* Berikut Teknologi *Firebase* yang digunakan dalam pembangunan Aplikasi perencanaan keuangan.

2.7.1 Firebase Cloud Messaging

Firebase Cloud Messaging adalah sebuah layanan yang digunakan untuk pemberitahuan (*notifications*) pada aplikasi berbasis Android, iOS maupun Web[6]. FCM memberikan kemudahan dalam menyampaikan pesan secara gratis tidak terikat besarnya suatu pesan selain itu FCM *services* akan mengatur setiap pesan yang dikirim agar sesuai dengan perangkat mobile tujuan pesan (*receiver*) [17].

Implementasi *Firebase Cloud Messaging* mencakup dua komponen utama untuk mengirim dan menerima pesan yaitu server aplikasi yang digunakan untuk membuat dan mengirim pesan serta yang menerima pesan seperti android. Seperti gambar dibawah ini.



Gambar 2.5 Cara Kerja *Firebase Cloud Messaging*

(Sumber : https://firebase.google.com/docs/cloud-messaging/10-April-2019)

Dengan *firebase cloud messaging* dapat mengirim pesan berupa pesan notifikasi dan juga pesan data. Pesan notifikasi digunakan jika ingin FCM menangani penampilan notifikasi atas nama aplikasi klien. Dan pesan data digunakan jika ingin memproses pesan pada aplikasi klien.

2.8 API

API atau *Application Programming Interface* adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan programmer saat membangun perangkat lunak. API memungkinkan programmer utuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API dapat menjelaskan cara sebuah tugas tertentu dilakukan dan menyertakan penjelasan dari fungsi atau rutin yang disediakannya. Tujuan penggunaan API yaitu untuk mempermudah dan mempercepat proses pembangunan suatu aplikasi dengan menyediakan function secara terpisah sehingga developer tidak perlu membuat fitur yang serupa. Berikut beberapa api yang digunakan dalam pembangunan aplikasi.

2.8.1 API Toko Online

Toko online atau yang di sebut *e-commerce* terdiri dari dua suku kata yaitu toko dan online. Toko merupakan sebuah tempat dengan bangunan yang nyata yang menjual barang-barang. Sedangkan online adalah keadaan dimana terhubung ke suatu jaringan yang besar. Toko online adalah tempat berlangsungnya aktifitas jual beli yang terhubung melalui suatu jaringan dalam hal ini internet. Proses menjual dan membeli barang atau disebut belanja online tanpa bertatap muka bisa dilakukan dimana saja dan kapan saja .

Tujuan dari penggunaan *Application Programming Interface* pada sebuah toko online yaitu agar developer tidak perlu membuat fitur yang sama yang ada pada sebuah toko online. Dengan menggunakan API Toko Online, aplikasi yang sedang dibuat dapat diintegrasikan ke toko online yang dituju dan dapat menggunakan fitur yang ada pada toko online.

2.8.2 API RajaOngkir

RajaOngkir menyediakan layanan API onglos kirim sehingga dapat membuat perhitungan ongkos kirim otomatis di toko online. Selain itu memberikan kemudahan bagi para penjual dan pembeli online untuk memeriksa ongkos kirim barang pesanan.

2.9 Google Cloud

Merupakan layanan komputasi awan yang berjalan di infrastruktur yang sama, yang digunakan google secara internal untuk produk penggunaan akhir, seperti Google penelusuran dan Youtube. Bersamaan dengan seperangkat alat manajemen, Google *cloud* menyediakan serangkaian layanan *cloud modular* termasuk komputasi, penyimpanan data, analisis data dan pembelajaran mesin. Untuk registrasi membuat akun google cloud membutuhkan detail kartu kredit. Google platform awan menyediakan layanan infrastruktur, layanan platform , dan lingkungan komputasi tanpa server. Berikut merupakan fitur yang ada dalam google cloud pada gambar 2.6 Dibawah ini

Gambar 2.6 Fitur Pada *Google Cloud*

(Sumber : https://Cloud.Google.com/10-April-2019)

2.9.1 Google Compute Engine

Merupakan komponen infrastruktur yang termasuk sebagai layanan dari *google cloud platform* yang dibangun diatas infrastruktur global yang menjalankan Google Search, Gmail, Youtube dan layanan lainnya. *Google Compute Engine* adalah sebuah jenis layanan *Cloud* yang ditawarkan oleh Google unuk membantu kita membangun sebuah *cloud* server, jad kita bsa mengontrol penuh mesin linux yang kita bangun diatas server Google. Berikut logo google *compute engine* pada gambar 2.7 dibawah ini.

Gambar 2.7 Logo *Google Compute Engine*

Layanan ini mampu menmpung aplikasi web kita yang dibangun dengan menggunakan bahasa pemrograman seperti java *Google Compute Engine* memungkinkan pengguna meluncurkan mesin virtual sesuai permintaan. Mesin virtual dapat diluncurkan dari gambar standar atau gambar khusus yang dibuat oleh pengguna.Pengguna *Google Compute Engine* harus diautentikasi berdasarkan OAuth 2.0 sebelum meluncurkan Mesin Virtual.Google Compute Engine dapat diakses melalui Konsol Pengembang, REST (*Representational State Transfer*), atau baris perintah antarmuka.

Secara umum ada tiga jenis layanan pada *cloud computing*[17]. Dimana pada ketiga arsitektur tersebut pengguna tidak mengatur secara langsung[17].

1. *Infrastructure as a Service* (*IaaS*). Menyediakan layanan sampai pada sistem operasi. Jadi pengguna dapat memilih sistem operasi yang akan digunakan dalam bentuk *virtual machine*.

2. *Platform as a Services* (*Paas*). Menyediakan layanan pada level platform, jadi pengguna tidak lagi direpotkan dengan instalasi sistem operasi, web server, database server dan aplikasi lainnya.
3. *Software as a Services* (*SaaS*). Menyediakan langsung kepada pengguna bentuk aplikasi yang sudah jadi seperti *Google docs*, *Office* 365, *Gmail*

2.10 MongoDB

Merupakan salah satu produk database noSQL *open source* yang menggunakan struktur data JSON untuk menyimpan datanya. Istilah ini dapat diartikan secara awam dengan non relasional karena berbeda dengan MySQL yang merupakan RDBMS (*relational database management system*). Berikut logo mongoDB pada gambar 2.8 dibawah ini

Gambar 2.8 Logo MongoDB

MongoDB sering dipakai untuk aplikasi berbasis *cloud, Grid computing*, atau *big data*. database noSQL dibagi menurut format penyimpanan dokumentnya. Berikut adalah pengelompokan database noSQL berdasarkan model penyimpanan data.

1. **Dokumen database** contohnya MongoDB, setiap satu object data disimpan dalam satu dokumen. Dokument sendiri bisa terdiri dari *key-value*, dan *value* sendiri bisa berupa *array* atau *key-value* bertingkat.
2. ***Graph*** Format penyimpanan data dalam struktur *graph*. Format *graph* digunakan untuk data yang saling berhubungan seperti jejaring sosial.

Contoh database noSQL dengan format ini adalah *Neo4J* dan *FlockDB. FlockDB* dipakai oleh twitter

3. ***Key – Value*** contoh database jenis ini adalah Apache Cassandra.
4. ***Object* Database**. Format database yang disimpan dalam object object, *Object* disini sama dengan pengertian object di Pemrograman beroreintasi *object* , Contoh databasenya adalah *Db4o*.

2.11 Web Service

Konsep teknologi *Web Service* muncul untuk mendukung sistem terdistribusi yang berjalan pada infrastruktur yang berbeda[19]. SOAP (*Simple Object Application Protocol*) dan beberapa teknologi yang didukung seperti WSDL (*Web Service Description Language*) dan UDDI (*Universal Description Discovery, and Integration*) merupakan kombinasi dari XML (*eXtensible Markup Language*) yang dikirimkan melalui HTTP (*Hyper Text Transport Protocol*)[19].

Sampai dengan saat ini teknologi *web service* terus berkembang. Salah satu teknologi yang populer saat ini adalah REST (*Representational State Transfer*) atau terkadang disebut dengan RESTful. Beberapa contoh *RESTful web service* adalah Amazon's Simple Storage Service (S3), Atom Publishing Protocol, dan Google Maps[20]. Pada prinsipnya *request* ke suatu RESTful *web service* sebenarnya adalah suatu *HTTP Request*[21].

Web service menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler. Beberapa alasan mengapa digunakannya *web service* adalah sebagai berikut:

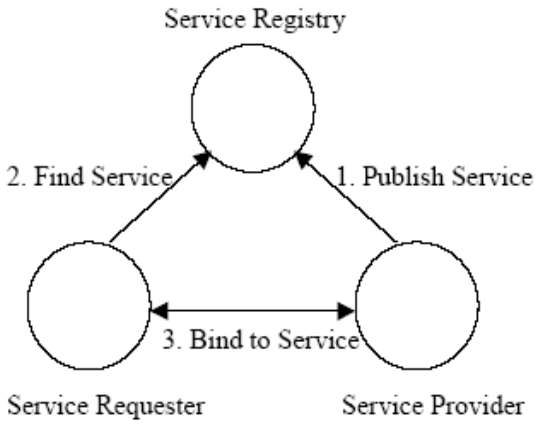
1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa *bisnis logic* atau *class* dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. *Web service* memiliki kemudahan dalam proses *deployment*-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web service* cukup

di-upload ke *web server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.

3. *Web service* berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian *web service* tidak memerlukan konfigurasi khusus di sisi *firewall.*

2.11.1 Arsitektur Web Service

Web service memiliki tiga entitas dalam arsitekturnya, yaitu *Service Requester* (peminta layanan), *Service Provider* (penyedia layanan), *Service Registry* (daftar layanan) . berikut merupakan arsitektur web service pada gambar 2.9 dibawah ini



Gambar 2.9 Arsitektur Web Service

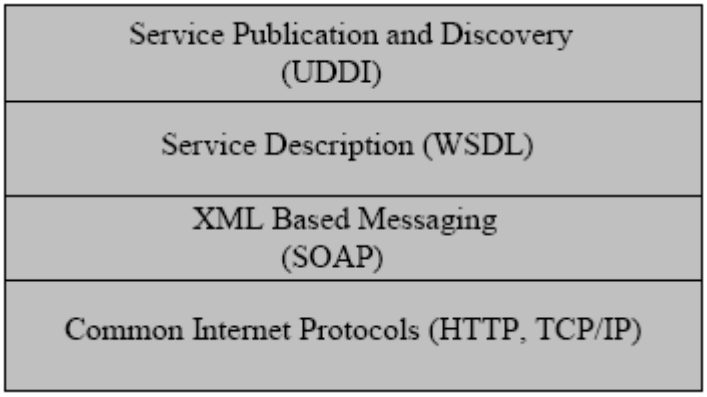
Berdasarkan gambar 2.9 diatas, maka dapat dijelaskan sebagai berikut :

1. *Service Provider*: Berfungsi untuk menyediakan layanan/*service* dan mengolah sebuah *registry* agar layanan-layanan tersebut dapat tersedia.
2. *Service Registry*: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/*service* yang telah di-register.
3. *Service Requestor*: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

Secara umum, *web service* memiliki tiga operasi yang terlibat di dalamnya, yaitu:

1. *Publish/Unpublish* Menerbitkan/menghapus layanan ke dalam atau dari registry.
2. *Find Service requestor* mencari dan menemukan layanan yang dibutuhkan.
3. *Bind Service* requestor setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke *service provider* untuk melakukan interaksi dan mengakses layanan/*service* yang disediakan oleh *service provider*.

2.11.2 Komponen Web Service



Gambar 2.10 Komponen *Web Service*

Web service secara keseluruhan memiliki empat layer komponen seperti pada gambar 2.10, yaitu:

1. Layer 1: Protokol internet standar seperti HTTP, TCP/IP
2. Layer 2: *Simple Object Access Protocol* (SOAP), merupakan protokol akses objek berbasis XML yang digunakan untuk proses pertukaran data/informasi antar layanan.
3. Layer 3: *Web Service Definition Language* (WSDL), merupakan suatu standar bahasa dalam format XML yang berfungsi untuk mendeskripsikan seluruh layanan yang tersedia.

2.12 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 JavaScript, 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 dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.JSON terbuat dari dua struktur:

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

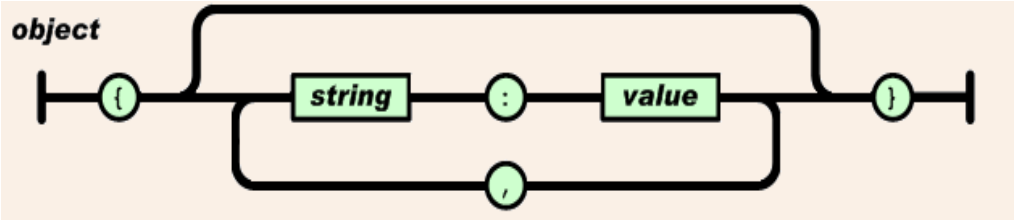
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut :

2.12.1 Bentuk JSON

JSON menggunakan bentuk sebagai berikut:

1. Objek

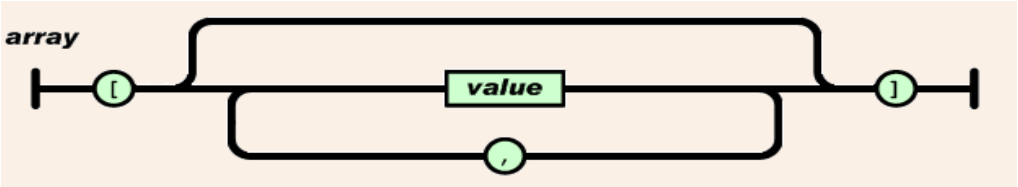
Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurang kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



Gambar 2.11 *Object* JSON

2. Larik

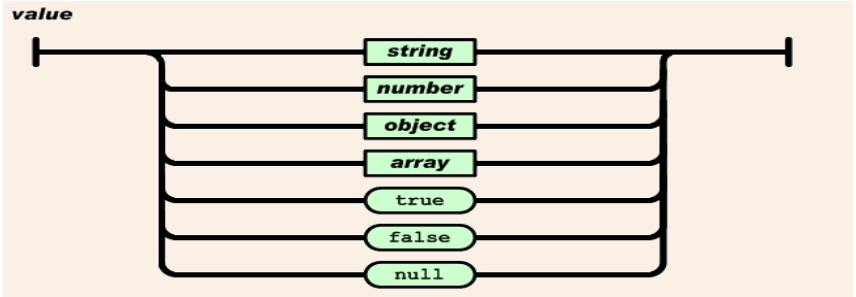
   Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



Gambar 2.12 *Array* JSON

3. Value

   Nilai (*value*) 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.

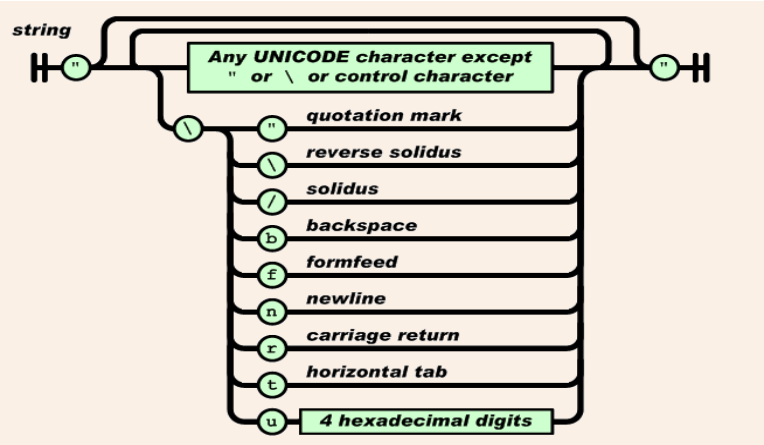


Gambar 2.13 *Value* JSON

4. ***String***

   String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes

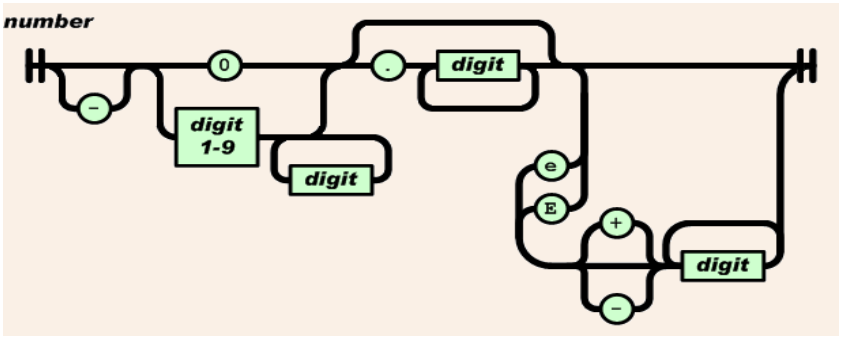
"\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan string C atau Java.



Gambar 2.14 *String* JSON

5. Angka

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



Gambar 2.15 Angka JSON

6. Spasi

   Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.13 Unified Model Language

Merupakan himpunan struktur dan teknik pemodelan desain program ber orientasi objek. UML adalah metodelogi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML juga merupakan bahasa standar untuk mendokumentasikan, menspesifikasi, dan membangun sistem perangkat lunak.

Analisis desain berorientasi objek atau disebut OOAD adalah metode analisis yang memerikasa requirements dari sudut pandang kelas-kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOAD merupakan cara untuk menjabarkan masalah menggunakan model yang dibuat menurut konsep kenyataan. Konsep OOAD mencangkup analisis dan desain sebuah sistem dengan pendekatan objek diantaranya :

1. *Object*

   Objek adalah benda yang ada di sekitar kita. Sebuah objek memiliki keadaan sesaat yang disebut state. State dari sebuah objek adalah kondisi dari objek atau himpunan keadaan yang menggambarkan objek tersebut. State dinyatakan dengan nilai dari atribut objeknya. Atribut adalah nilai internal suatu objek yang mencerminkan karakteristik objek, kondisi sesaat, koneksi dengan objek lain dan identitas.

2. *Class*

   Class adalah himpunan objek yang sejenis yaitu mempunyai atribut, dan ber-relasi umum dengan objek lain dan semantik umum. *Class* adalah abstraksi dari objek dalam dunia nyata. *Class* menetapkan spesifikasi perilaku dan atribut dari objek tersebut.

3. *Black Box*

   Sebuah objek merupakan kotak hitam, konsep tersebut merpakan dasar dari implementasi objek. *Black box* berisi kode dan data, prosesnya yaitu enkapsulasi dan *message*.

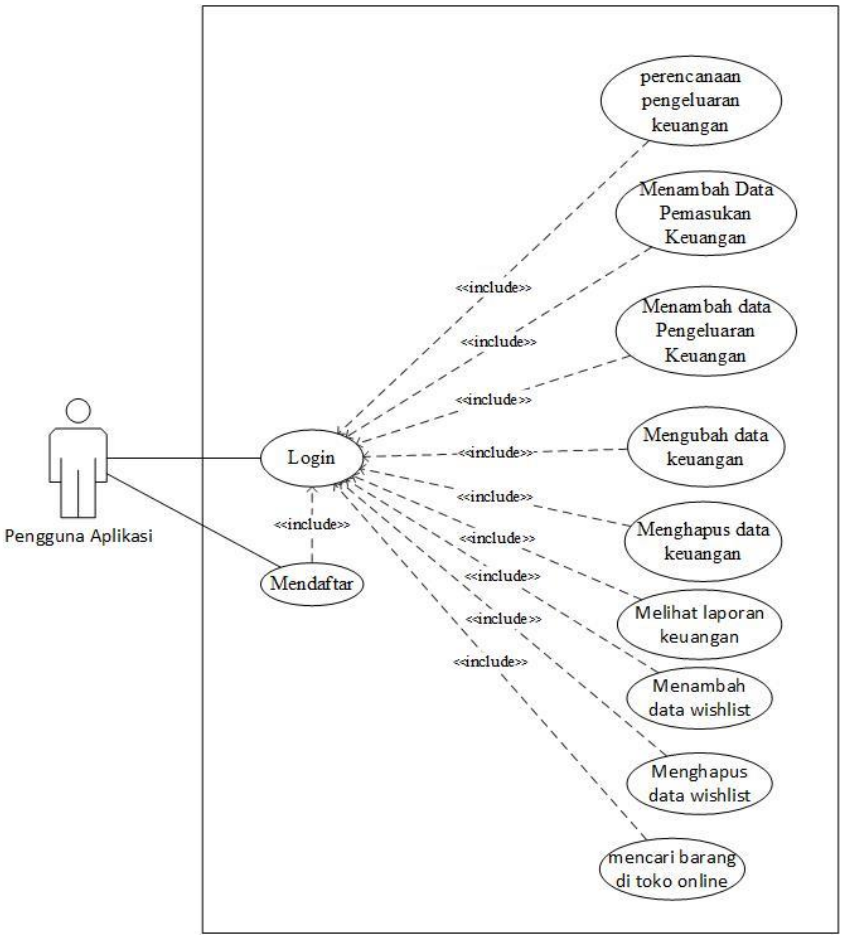
4. Asosiasi dan Agregasi

   Asosiasi adalah hubungan yang berkaitan antara sejumlah objek. Sedangkan agregasi adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada suatu objek merupakan bagian dari objek yang lain

UML disebut sebagai bahasa pemodelan bukan metode. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Berikut ini merupakan beberapa bagian dari UML adalah sebagai berikut :

1. *Use Case Diagram*

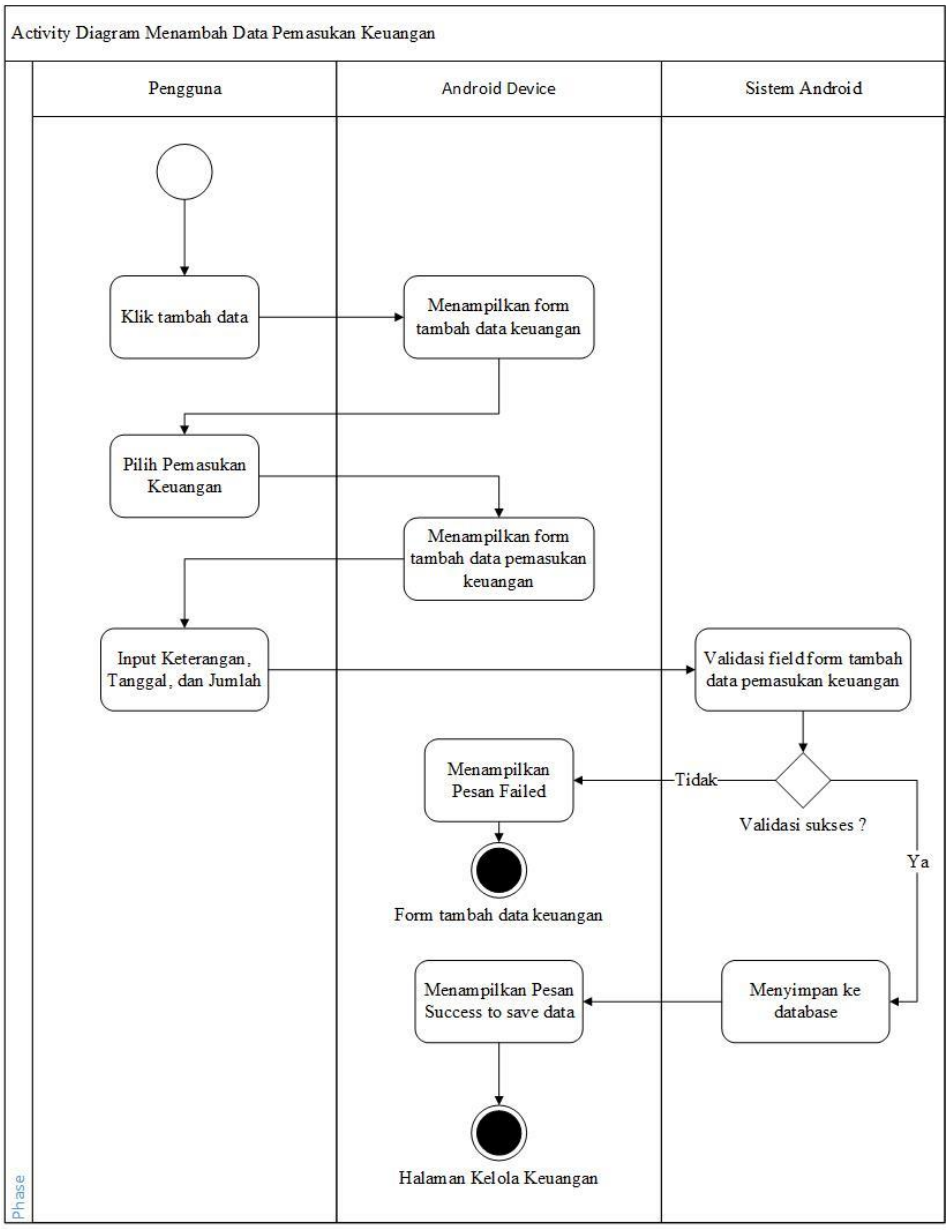
   Merupakan abstraksi interaksi antara aktor dan sistem. Berkut adalah contoh use case diagram.



Gambar 2.16 Contoh *Use Case* Diagram

2. *Activity* Diagram

Activity diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case* atau interaksi. Berikut merupakan contoh *Activity* Diagram.



Gambar 2.17 Contoh *Activity* Diagram

3. *Use Case* Scenario

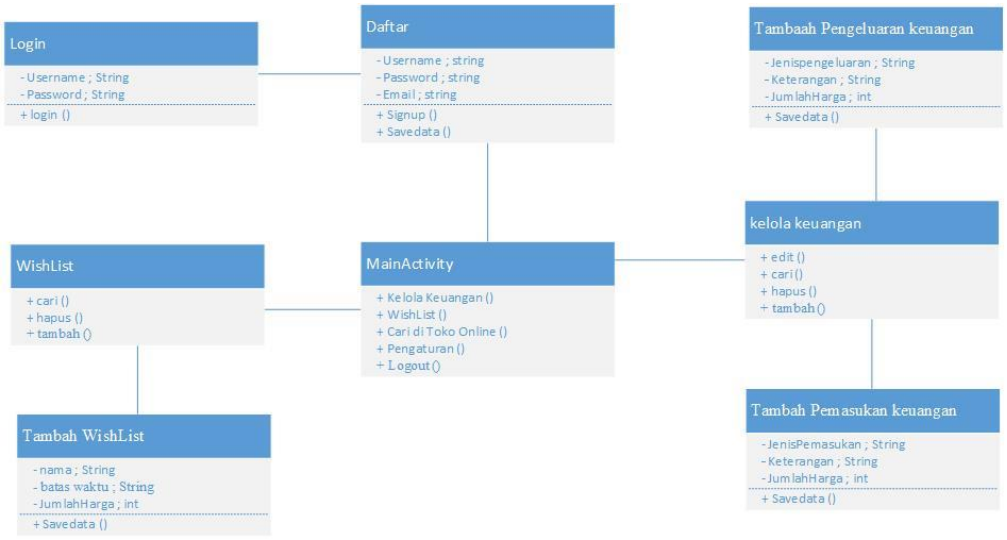
skenario *use case* / *use case* skenario adalah alur jalannya proses *use case* dari sisi aktor dan system. Skenario normal adalah skenario bila system berjalan normal tanpa terjadi kesalahan atau error. Sedangkan skenario alternatif adalah skenario bila system tidak berjalan normal atau mengalami error. Skenario normal dan skenario alternatif dapat berjumlah lebih dari satu. Berikut contoh *use case* scenario.

Tabel 2.1 Contoh *Use Case* Scenario

Requirement A.2		
Data log-in terdiri dari ID dan password		
Use Case Name	Log-in	
Related Requirements	Requirement A.2	
Goal Context	Melakukan login ke dalam perangkat lunak	
Pre – conditions	Form Login Perangkat lunak ditampilkan	
Successful End Condition	Menampilkan activity utama perangkat lunak	
Failed End Conditions	Menampilkan pesan kesalahan log-in	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol login	
Main Flow	Step	Action
	1	Sistem menampilkan form login perangkat lunak
	2	Pengguna mengisi data Login
	3	Pengguna menekan tombol log-in
	4	Sistem melakukan validasi field login
	5	Sistem melakukan autentikasi data login dengan database
	6	Sistem menampilkan pesan berhasil log-in
	7	Sistem menampilkan halaman utama
Extension	Step	Branching Action
	4.1	ID dan Password kosong
	4.2	ID dan Password Terisi
	6.1	Pesan gagal log-in
	6.2	Pesan berhasil log-in

4. *Class* Diagram

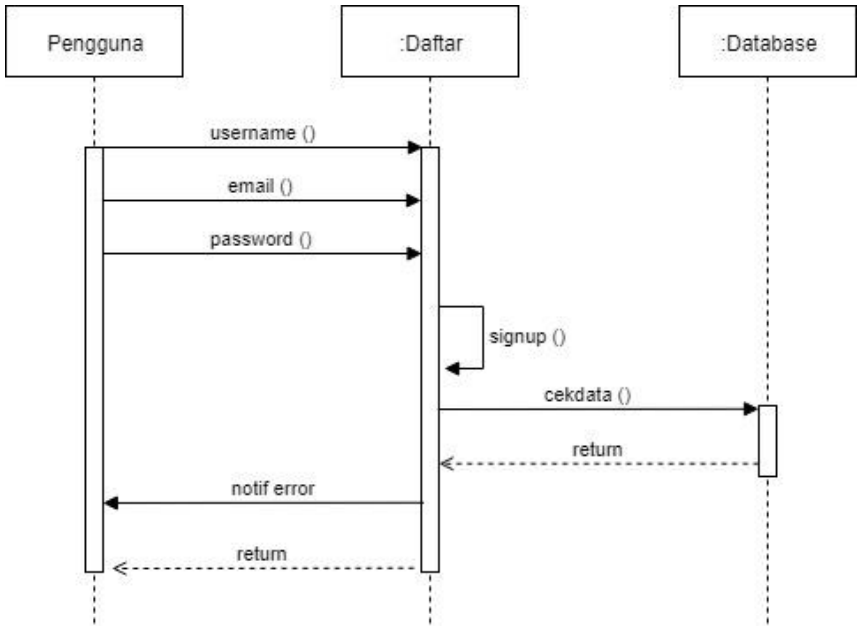
   Merupakan gambaran struktur dan deskripsi *class*, *package*, dan objek serta hubungannya antara satu sama lain.



Gambar 2.18 Contoh *Class* Diagram

5. *Sequence* Diagram

   Sequence diagram digunakan untuk menggambarkan prilaku pada sebuah skenario. Berikut contoh sequence diagram.



Gambar 2.19 Contoh *Sequence* Diagram

2.14 Android Studio

Merupakan *Integrated Development Enviroment* (IDE) untuk sistem operasi Android, yang dibangung diatas perangkat lunak *JetBrains IntelliJ* IDEA dan didesain khusus untuk pengembangan Android. IDE ini merupakan pengganti dari *Eclipse Android Development Tools* (ADT) yang sebelumnya merupakan IDE utama untuk pengembangan aplikasi android. android studio memiliki beberapa keungulan diantaranya :

1. Editor kode yang cerdas, android studio membantu untuk membuat kode dengan cepat, dengan fitur *intelligent code editor* yang memberikan kemudahan dalam menganalisis kode dan menyediakan saran kode yang akan digunakan dengan sistem *auto complete*.
2. Android studio menyediakan alat GUI untuk mempermudah developer dalam melihat dan merancang aplikasi.
3. Dioptimalkan untuk seluruh perangkat android
4. *Firebase assistant* membantu menghubungkan aplikasi dengan *Firebase* dan menambahkan layanan seperti *Analytics*, Autentikasi, Notifikasi, dan lainnya dengan prosedur sesuai dengan urutan di dalam Android Studio.

2.15 Flutter

Fluttter adalah SDK untuk pengembangan aplikasi mobile yang dikembangkan oleh google. Sama seperti *react native*, *framework* ini dapat digunakan untuk membuat atau mengembangkan aplikasi mobile yangdapat berjalan pada iOS dan Android. Berikut logo flutter pada gambar 2.20

Gambar 2.20 Logo Flutter

Dapat dibuat menggunakan Bahasa C, C++, Dart dan Skia. Semua kodenya di compile dalam kode native nya (Android NDK, LLVM, AOT *Compiled*). Berikut adalah langkah instalasi flutter

1. Download flutter di https://flutter.dev
2. Ekstrak berkas zip/rar flutter yang sudah di download
3. Buka android studio yang sudah terpasang, tunggu hingga muncul tampilan halaman awal android studio
4. Buka *configure*, lalu pilih *plugins*
5. Setelah muncul jendela plugins, pilih *browse repository*, kemudian lakukan pencarian dengan kata kunci flutter
6. Kemudian install flutter dan restart android studio
7. Selanjutnya dapat memulai dengan start a new flutter project

Versi pertama Flutter dikenal sebagai "*Sky*" dan berjalan pada sistem operasi Android. Diresmikan pada perhelatan *Dart developer summit* tahun 2015, dengan tujuan untuk mampu merender grafis secara konsisten pada 120 bingkai per detik. Komponen utama Flutter termasuk :

1. ***Flutter engine***

   Flutter engine, ditulis terutama dengan bahasa pemrograman C++, memberikan dukungan rendering tingkat rendah menggunakan *library* grafik Skia milik Google. Selain itu, *flutter engine* juga berinteraksi dengan perkakas pengembangan perangkat lunak (*SDK*) spesifik-serambi (*flatform-specific*) seperti yang disediakan oleh Android dan iOS.

2. ***Foundation library***

   Foundation library, ditulis dengan bahasa pemrograman Dart, menyediakan fungsi dan *class-class* dasar yang digunakan untuk membangun aplikasi menggunakan Flutter, seperti *API* untuk berkomunikasi dengan *engine*.

3. **Widget spesifik desain**

   Framework Flutter berisi dua set widget yang disesuaikan dengan bahasa desain tertenu. Widget *Material Design* menerapkan bahasa desain Google dengan nama yang sama, sedangkan widget 'Cupertino' meniru desain iOS milik Apple.

   (sumber : https://id.wikipedia.org/wiki/Flutter_(perangkat_lunak)