BAB 2
TINJAUAN PUSTAKA

2.1 Covid-19

Covid-19 merupakan sejenis virus dari famili Coronaviridae yang berimplikasi terhadap penyakit menular dan mematikan yang menyerang mamalia seperti manusia pada saluran pernafasan hingga ke paru-paru. Virus ini bisa menyerang siapa saja, baik bayi, anak-anak, orang dewasa, maupun lansia. Pada umumnya pengidap Covid-19 akan mengalami gejala awal berupa demam, sakit tenggorokan, pilek, batuk-batuk bahkan dapat menyebabkan pneumonia hingga kematian [15].

Virus ini awalnya ditemukan di Kota Wuhan, Cina pada akhir Desember 2019. Virus ini menular dengan cepat dan menyebar di berbagai wilayah lain di Cina bahkan ke beberapa negara termasuk Indonesia. Menurut WHO virus corona Covid-19 dapat menular melalui kontak langsung dalam jarak dekat dengan pengidap Covid-19 melalui cairan/tetesan kecil dari hidung atau mulut yang keluar dari tubuh penderita pada saat batuk, menghembuskan nafas atau mengeluarkan ludah dan riyak [16].

Berdasarkan dokumen Kemenkes RI Nomor HK.01.07/MENKES/413/2020 tentang Pedoman Pencegahan dan Pengendalian Coronavirus Disease 2019 (COVID-19), virus ini bisa mati dalam rentang waktu 5-7 hari, masa inkubasinya paling pendek berlangsung selama dua sampai tiga hari. Sedangkan paling lama bisa mencapai 10-12 hari. Ini adalah rentang waktu yang dibutuhkan oleh virus untuk menjangkit dan menampakkan gejala-gejala awal. Dalam masa ini virus corona sulit untuk dideteksi. Virus corona sangat sensitif terhadap panas dengan suhu setidaknya 56°C selama 30 menit. Virus corona belum bisa diobati dengan penanganan medis apa pun. Namun dengan sistem imun tubuh yang cukup baik, virus corona tak mudah menyebar ke seluruh anggota tubuh.

2.2 Isolasi Mandiri

Isolasi mandiri (isoman) merupakan upaya mencegah penyebaran Covid-19 dengan cara berdiam diri baik di rumah, tempat kos, hotel, apartemen atau di tempat khusus isoman yang disediakan Pemerintah sambil memantau kondisi diri seraya

mengurangi aktivitas sosial dengan tetap menjaga jarak aman dari orang sekitar atau keluarga. Pasien yang perlu melakukan isoman adalah pasien dengan gejala ringan seperti demam, batuk, pilek, nyeri tenggorokan, atau gejala penyakit pernafasan lainnya dan pasien tanpa gejala yang pernah kontak langsung dengan terduga penderita Covid-19 atau memiliki riwayat perjalanan ke daerah zona merah [6].

Isoman bagi pasien yang memiliki gejala ringan dilakukan selama 10 hari ditambah 3 hari bebas demam dan gejala pernapasan, sedangkan bagi pasien yang tidak memiliki gejala disarankan untuk isoman selama 10 hari [3]. Dalam surat edaran yang dikeluarkan oleh Kemenkes RI Nomor HK.02.01/MENKES/18/2022 tentang Pencegahan serta Pengendalian Kasus Covid-19, pasien yeng terkonfirmasi Covid-19 gejala ringan dan tanpa gejala diperbolehkan melakukan isoman jika sudah memenuhi syarat klinis. Adapun untuk syarat klinisnya yaitu sebagai berikut:

1. Berusia di bawah 45 tahun.
2. Tidak memiliki komorbid.
3. Bisa akses telemedicine maupun layanan kesehatan lainnya.
4. Berkomitmen untuk melalukan isoman sebelum diizinkan keluar.

Saat melakukan isoman ada beberapa hal yang boleh dan tidak boleh dilakukan diantaranya berbagi alat makan, alat mandi dan pakaian bersama anggota keluarga lainnya (apabila berbagi kamar mandi atau mesin cuci pakaian hendaknya membersihkannya dengan desinfektan terlebih dahulu setelah menggunakannya), menjaga jarak 1 meter dari keluarga, menerapkan perilaku sehat dan bersih, tetap menjaga pertahanan tubuh dengan menerapkan pola hidup sehat yaitu makan makanan bergizi seimbang, perbanyak makan buah dan sayur, istirahat yang cukup, melakukan olahraga ringan dan hindari merokok ataupun minuman beralkohol [17].

2.3 Detak Jantung

Detak jantung adalah jumlah banyaknya jantung berdetak dalam satu waktu. Jantung berdetak bertujuan untuk memompa darah yang telah bersih dari ventrikel kiri ke seluruh pembuluh darah tubuh melalui aorta. Pemompaan ini menyebabkan darah menekan dinding arteri, sehingga menciptakan gelombang tekanan yang biasa disebut denyut nadi. Denyut nadi ini dapat diraba/palpasi untuk menilai kecepatan detak jantung, ritme dan fungsinya.

Pada umumnya, detak jantung diukur dalam kurun waktu satu menit, sehingga detak jantung memiliki satuan beats per minute (bpm). Detak jantung dapat berubah-ubah tergantung dari aktivitas manusia. Terdapat beberapa faktor yang dapat mempengaruhi detak jantung, yaitu temperatur udara, posisi tubuh, emosi, ukuran tubuh, dan penggunaan obat. Selain itu detak jantung juga berbeda-beda pada tiap rentang usia. Tabel 2.1 berikut menunjukkan detak jantung pada saat aktivitas normal (istirahat) berdasarkan rentang usia menurut National Institute of Health (NIH) [18].

Tabel 2.1 Detak Jantung Istirahat Berdasarkan Usia

Usia	**Detak Jantung**		
	Minimal (bpm)	**Maksimal (bpm)**	**Rata-Rata (bpm)**
< 1 Bulan	70	190	130
1 Bulan – 11 Bulan	80	160	120
1 Tahun – 2 Tahun	80	130	105
3 Tahun – 4 Tahun	80	120	100
5 Tahun – 6 Tahun	75	115	95
7 Tahun – 9 Tahun	70	110	90
≥ 10 Tahun	60	100	80

Bagian tubuh yang dapat digunakan untuk mengukur detak jantung yaitu pergelangan tangan, dibawah alis, sisi leher, dan diatas telapak kaki. Namun pada umumnya pergelangan tangan menunjukkan hasil yang lebih akurat. Durasi yang cukup untuk mendapatkan hasil pengukuran detak jantung yang akurat adalah selama 30 detik. Hal-hal yang perlu diperhatikan saat pemeriksaan detak jantung yaitu [18]:

1. Kecepatan, kecepatan detak jantung terbagi menjadi tiga yaitu:

a. Bradycardia: detak jantung lambat (<60 bpm).

b. Tachycardia: detak jantung cepat (>100 bpm).

c. Normal: detak jantung normal (60-100 bpm) pada orang dewasa.

2. Irama, irama detak jantung terbagi menjadi dua yaitu:

a. Regularly irregular: dijumpai pola dalam iregularitasnya.

b. Irregularly irregular: tidak dijumpai pola dalam iregularitasnya, terdapat pada fibrilasi atrium.

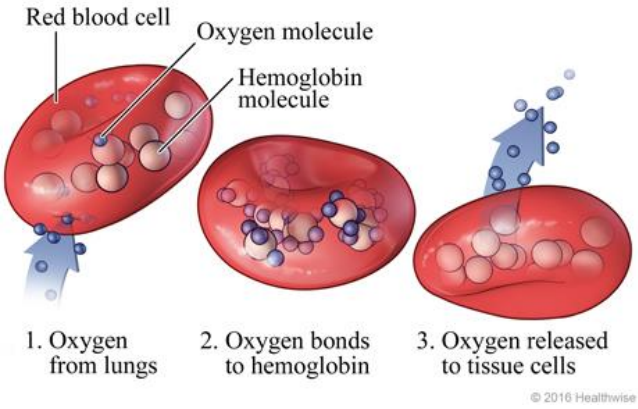
3. Volume Nadi, volume nadi dapat dinilai dari tiga buah aspek yaitu:

a. Volume nadi kecil: tahanan terlalu besar terhadap aliran darah, darah yang dipompa jantung terlalu sedikit (pada efusi perikardial, stenosis katup mitral, payah jantung, dehidrasi, syok hemoragik).
b. Volume nadi yang berkurang secara lokal: peningkatan tahanan setempat.
c. Volume nadi besar: volume darah yang dipompakan terlalu banyak, tahanan terlalu rendah (pada bradycardia, anemia, hamil, hipertiroidisme).

2.4 Saturasi Oksigen

Saturasi oksigen adalah presentasi hemoglobin yang berikatan dengan oksigen dan dibandingkan dengan total hemoglobin dalam darah. Hemoglobin dalam sel darah merah bertugas sebagai media transport oksigen dari paru-paru ke jaringan diseluruh tubuh dan membawa kembali karbondioksida sisa metabolisme dari seluruh tubuh ke paru-paru.

Dalam kondisi fisiologis normal presentase O_2 yang diangkut oleh hemoglobin dalam darah hampir 98%. Kondisi saat hemoglobin (Hb) berikatan dengan O_2 akan terbentuk oksihemoglobin (HbO_2) seperti pada Gambar 2.1 berikut [19].



Gambar 2.1 Proses Terbentuk Oksihemoglobin (HbO2)

Sumber: https://www.upmc.com/health-library/article?hwid=tp10337/

Dalam kondisi normal saturasi oksigen dalam darah manusia berkisar antara 95-100%. Namun terdapat toleransi medis hingga 91% walau sudah di anggap rendah. Apabila kadar saturasi oksigennya kurang dari 90% maka di sebut hipoksemia. Tingkatan kadar saturasi oksigen dalam darah manusia dapat dilihat pada Gambar 2.2 berikut [20].

Table 4. Pulse Oximetry: What Do the Numbers Mean?[7]

SpO_2, %	PaO_2, mm Hg	Oxygenation Status
95-100	80-100	Normal
91-94	60-80	Mild hypoxia
86-90	50-60	Moderate hypoxia
Less than 85	Less than 50	Severe hypoxia

Gambar 2.2 Tingkatan Kadar Saturasi Oksigen

Sumber: https://www.npjournal.org/article/S1555-4155(07)00210-3/fulltext

2.5 Silent Hypoxia

Silent hypoxia atau yang kerap disebut happy hypoxia adalah keadaan dimana kadar saturasi oksigen dalam tubuh lebih rendah dari kondisi normal tetapi penderita tidak merasakan gejala apapun. Hipoksia merupakan kondisi berbahaya karena dapat mengganggu fungsi otak, hati, dan organ lainnya. Kondisi ini dapat terjadi jika terdapat gangguan dalam sistem transportasi oksigen dari mulai bernapas hingga oksigen tersebut digunakan oleh sel tubuh. Terlebih lagi pasien yang mengalami silent hypoxia tidak merasakan gejala akan terjadinya hipoksia. Karena pasien silent hypoxia cenderung tidak memiliki gejala, maka salah satu indikator yang dapat dilihat adalah tingkat saturasi oksigen. Pasien silent hypoxia hanya memiliki kadar saturasi oksigen 70-80% bahkan ada yang dibawah 50%, itu sangat rendah jika dibandingkan dengan kondisi kadar saturasi oksigen normal yaitu 94-100% [21].

Mengingat pasien dengan Covid-19 menunjukkan beberapa temuan yang tidak biasa, ada kemungkinan virus tersebut memiliki efek istimewa pada sistem kontrol pernapasan. ACE 2 (angiotensin-converting enzyme 2), reseptor sel SARS-CoV-2, virus yang bertanggung jawab untuk COVID-19, diekspresikan dalam badan karotid, tempat kemoreseptor merasakan oksigen. Reseptor ACE2 juga diekspresikan di mukosa hidung. Anosmia-Hiposmia terjadi pada dua pertiga pasien dengan Covid-19, dan olfactory bulb menyediakan sebuah bagian bersama coronavirus tertentu yang masuk ke otak. SARS-CoV-2 memperoleh akses ke otak melalui olfaktorius dan berkontribusi pada hubungan antara anosmia-hiposmia dan dispnea dan reseptor ACE2 berperan dalam respons dispnea depresi pada COVID-19 yang masih harus ditentukan.

Sains mengaitkan silent hypoxemia dengan perkembangan trombus di dalam pembuluh darah paru-paru. Peningkatan trombogenesis telah dicatat pada pasien dengan Covid-19. Trombus dalam pembuluh darah paru dapat menyebabkan hipoksemia berat, dan dispnea berhubungan dengan obstruksi pembuluh darah paru dan konsekuensinya. Dispnea juga bisa timbul dari pelepasan histamin atau stimulasi reseptor juxtacapillary di dalam pembuluh darah paru. Namun, tidak ada mekanisme biologis dimana trombus di pembuluh darah paru-paru menyebabkan dispnea tumpul (menghasilkan silent hypoxemia) [20].

2.6 Suhu Tubuh

Suhu tubuh merupakan keseimbangan antara produksi dan pengeluaran panas dari tubuh yang diukur dalam unit panas yang disebut derajat. Suhu tubuh manusia adalah panas atau dingin. Adapun faktor-faktor yang mempengaruhi suhu tubuh antara lain:

a. Faktor usia

Faktor usia sangat mempengaruhi suhu tubuh. Suhu tubuh anak-anak nilainya bervariasi dibandingkan dengan suhu tubuh orang dewasa atau masa remaja. Sebagian lansia terutama yang berusia diatas 70 tahun beresiko mengalami hipotermia (suhu tubuh rendah dibawah 36°C)

b. Faktor hormon

Melakukan aktivitas yang bervariasi seperti olahraga dapat merangsang hormon yang berada pada tubuh meningkat sebesar 0,6°C.

c. Faktor Pikiran

Orang yang mengalami stres atau kecemasan akan mengalami peningkatan suhu karena metabolisme dalam tubuh sedang tidak stabil.

Pengukuran suhu tubuh dapat dilakukan dengan menggunakan termometer. Pengukuran dengan termometer ini dapat dilakukan pada ketiak, mulut, dahi atau dubur. Namun pelu diketahui bahwa pengukuran yang dilakukan pada masing-masing bagian tubuh tersebut akan memberikan hasil yang berbeda. Jika pengukuran melalui dubur menunjukan suhu 38°C, maka pengukuran di mulut biasanya menunjukan 37,7°C dan di ketiak 37,5°C.

Suhu tubuh kritis bayi atau anak-anak adalah 38,3°C dan suhu tubuh orang dewasa adalah 39,4°C. Bila suhu tubuh bayi atau anak-anak di atas 38°C dan orang dewasa melebihi 39,4°C maka harus segera diperiksa kondisi kesehatanya, hal ini dikarenakan suhu tubuh yang tinggi baik pada bayi, anak-anak maupun orang dewasa menandakan bahwa kondisi kesehatanya sedang terganggu. Suhu tubuh manusia dibagi menjadi empat kategori yaitu dapat dilihat pada Tabel 2.2 berikut.

Tabel 2.2 Suhu Tubuh Manusia

Kategori	Suhu
Hipotermia	<36°C
Normal	36°C – 37,5°C
Pireksia	<37,5°C - 40°C
Hipertemia	>40°C

Adapun Penjelasan dari kempat kategori tersebut yaitu sebagai berikut:

1. Hipotermia adalah peningkatan suhu tubuh sehubung dengan tidak mampuh tubuh untuk meningkatkan pengeluaran panas atau menurunkan produksi suhu tubuh panas.
2. Normal adalah suhu tubuh manusa yang tergolong dalam kondisi normal yaitu 36°C – 37,5°C.
3. Pireksia adalah dimana manusia mengalami demam, demam memiliki 3 bagian, demam rendah, demam sedang dan demam tinggi. Suhu demam rendah 37,7°C – 38,8°C demam sedang 38,8°C - 40°C dan demam tinggi adalah >40°C.
4. Hipertemia adalah suatu kondisi suhu tubuh meningkat drastis dari suhu normal, hipertemia terjadi ketika sistem suhu tubuh tidak mampu menahan suhu panas dari lingkungan sekitar. Suhu tubuh lebih dari 40°C sudah dipastikan hipertemia.

2.7 Wearable Device

Wearable device disebut sebagai penggabungan beberapa teknologi, seperti teknologi elektronika dan komputer yang dikemas sedemikian rupa sehingga dapat dipakai atau dikenakan [22]. Biasanya wearable device ini disematkan ke dalam pakaian atau dibentuk menyerupai aksesoris, seperti gelang, jam tangan, lensa kontak, e-tekstil, kain pintar, topi, perhiasan anting, cincin, dan berbagai bentuk lainnya sehingga lebih mudah untuk dikenakan ke bagian tubuh pengguna [23].

Secara umum, wearable device atau wearable sensor memiliki sejumlah keunggulan dibandingkan dengan perangkat atau sensor yang tidak dapat dikenakan. Keunggulan tersebut seperti: dapat ditempatkan secara dinamis, perangkat dapat dinyalakan ketika waktu pengukuran diperlukan, dapat dirancang sesuai dengan parameter yang diukur, dirancang dengan ukuran yang kecil sehingga meminimalkan gangguan bagi pemakainya [24].

Wearable device tentunya tidak terlepas dari peranan teknologi sensor, komputasi, dan komunikasi. Sensor diperlukan untuk melakukan penginderaan terhadap lingkungan dari pengguna wearable device tersebut. Perangkat komputasi seperti komputer atau mikrokontroler diperlukan untuk mengolah data yang dihasilkan oleh sensor tersebut. Sedangkan sarana komunikasi diperlukan untuk mengirimkan data yang dihasilkan dari sensor. Sarana komunikasi menjadi hal yang sangat penting dalam wearable device terutama komunikasi secara nirkabel, apalagi kalau wearable device tersebut digunakan sebagai alat untuk melakukan monitoring [25]. Oleh karena itu, wearable device dapat diterapakan di berbagai bidang seperti: medis, olahraga, hiburan, hingga pertanian.

Pada penelitian ini wearable device yang digunakan berjenis smartband yang dapat dilihat pada Gambar 2.3 berikut.

Gambar 2.3 Xiaomi Mi Band 4

Sumber: https://www.mi.co.id/id/buy/product/mi-smart-band-4

Smartband ini sudah memiliki sensor heart rate dan SpO2, sehingga sudah cukup memenuhi kebutuhan penelitian untuk digunakan sebagai alat bantu monitoring kesehatan pasien isoman Covid-19. Adapun spesifikasi lengkap dari Smartband yang digunakan dapat dilihat pada Tabel 2.3 berikut.

Tabel 2.3 Spesifikasi Smartband

Nama	Xiaomi Mi Band 4
Layar	Layar sentuh AMOLED, 0.95 inci, 240 x 120 piksel (RGB)
Baterai	Tahan hingga 20 hari, Li-Polymer, 135 mAh
Konektivitas	Bluetooth 5.0 BLE
RAM, ROM	512 MB, 16 MB
Bobot	22,1 gram
Sensor	Accelerometer 3-sumbu + giroskop 3-sumbu; Sensor detak jantung PPG; Sensor kedekatan kapasitif
Sertifikasi	5 ATM
Bahan strap	Poliuretan termoplastik
Dukungan OS	Android 4.4, iOS 9.0 atau lebih tinggi

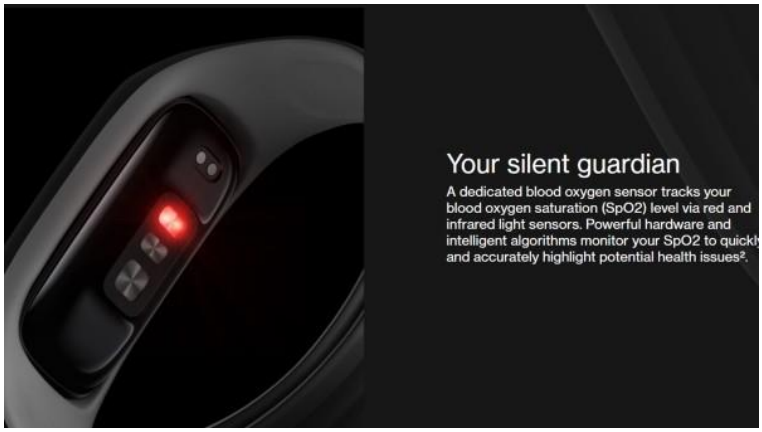
Sensor heart rate pada smartband menggunakan lampu LED hijau yang dipasangkan dengan fotodioda yang peka terhadap cahaya untuk mendeteksi jumlah darah yang mengalir di pergelangan tangan. Saat darah mengalir di pergelangan tangan akan memantulkan cahaya, dan cahaya tersebut akan dibaca oleh fotodioda, darah yang tidak memantulkan cahaya berarti volume darah lebih tinggi. Melalui pantulan cahaya tersebut, sensor akan menganalisa data dan menghitung nilai heart rate. Dengan mengedipkan lampu LED ratusan kali perdetik, smartband dapat menghitung frekuensi jantung berdenyut. Proses pengambilan detak jantung menggunakan smartband dapat dilihat pada Gambar 2.4 berikut.

Gambar 2.4 Pengambilan Detak Jantung Menggunakan Smartband

Sumber: https://gadgetren.com/wp-content/uploads/2020/09/MiBand5-Sensor-700x525.jpg

Sensor SpO2 pada smartband menggunakan LED dengan panjang gelombang cahaya yang berbeda, satu merah dan satu inframerah. Ini karena penyerapan

cahaya berbeda antara darah yang memiliki kadar oksigen normal dan darah yang kekurangan oksigen. Darah dengan oksigen normal menyerap lebih banyak cahaya inframerah, sementara darah yang kekurangan oksigen akan melewatkan cahaya tersebut. Sehingga bisa didapat kadar oksigen dalam darah berdasarkan efektifitas penyerapan cahaya yang dilepaskan dan diterima kembali oleh sensor ini. Proses pengambilan saturasi oksigen menggunakan smartband dapat dilihat pada Gambar 2.5 berikut.



Gambar 2.5 Pengambilan Saturasi Oksigen Menggunakan Smartband

Sumber: https://www.gizbot.com/img/600x90/img/2021/05/oneplussmartband-1621223375.jpg

2.8 Bluetooth Low Energy (BLE)

Bluetooth Low Energy atau disingkat sebagai BLE merupakan hasil pengembangan dari teknologi Bluetooth versi 4.0 yang dimana perangkat tidak perlu saling pairing agar dapat terkoneksi dan saling bertukar data. Dengan hasil pengembangan ini, para pengguna dapat memanfaatkannya untuk berbagai kebutuhan, seperti menggunakan BLE untuk menghubungkan smartphone dengan peralatan rumah, kendaraan, dan lain – lainnya.

BLE ini beroperasi pada frekuensi 2.4GHz yang memiliki karakteristik propagasi dalam ruangan yang sama sebagai penerima WiFi 2.4GHz. Sesuai dengan namanya, BLE ini memakan daya yang lebih sedikit dibandingkan dengan Bluetooth biasa. Namun BLE memiliki kekurangan yaitu tidak cocok digunakan apabila digunakan untuk menyambungkan dengan perangkat yang mengirim data terus menerus tanpa berhenti seperti speaker Bluetooth dan headset Bluetooth.

Berbeda dengan Bluetooth pada umumnya yang dapat menangani banyak data tetapi menghabiskan banyak daya dengan cepat dan juga biaya yang lebih banyak. Bluetooth Low Energy digunakan untuk aplikasi yang tidak perlu bertukar data dalam jumlah besar, karena itu dapat berjalan dengan daya baterai yang tahan lama dan biaya yang lebih murah [26].

2.9 Aplikasi

Secara istilah aplikasi merupakan suatu program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju. Pada saat ini, aplikasi telah berkembang menjadi beberapa platform. Terdapat tiga platform utama pada pengembangan sebuah aplikasi, yaitu aplikasi berbasis mobile, web dan desktop. Secara umum, aplikasi dibagi menjadi tiga macam aplikasi yaitu [27]:

1. Native Apps

Native apps adalah aplikasi yang dibangun secara spesifik untuk operating sistem tertentu. Jika aplikasi ini dibangun untuk sistem operasi iOS, maka aplikasi tersebut tidak akan dapat berjalan pada sistem operasi yang lain. Keuntungan utama dari jenis aplikasi ini adalah performanya yang tinggi serta memiliki user experience yang baik karena developer mengembangkan aplikasi ini menggunakan UI dari perangkat native. Secara keseluruhan, aplikasi ini menawarkan pengalaman pengguna yang lebih baik karena dapat bekerja dengan cepat, responsive, serta didistribusikan melalui app store.

2. Web Apps

Web apps berjalan menggunakan browser dan biasanya ditulis dalam HTML5, JavaScript atau CSS. Karena aplikasi menargetkan browser, maka web app dapat berjalan pada berbagai sistem operasi. Selain itu, aplikasi ini juga didesain secara responsive sehingga tampilan web dapat menyesuaikan ukuran layar pada perangkat yang digunakan oleh user.

3. Hybrid Apps

Hybrid apps dapat dikatakan seperti kombinasi dari dua macam aplikasi, yaitu native dan web. Hybrid apps memiliki dua bagian utama. Bagian pertama adalah

kode back-end, dan yang kedua adalah native shell yang dapat diunduh dan memuat kode menggunakan tampilan web. Hybrid apps dinilai lebih mudah dan cepat untuk dikembangkan dibanding dengan native apps. Namun kecepatan hybrid apps bekerja lebih lambat daripada aplikasi native karena bergantung pada kecepatan browser user.

2.10 Android

Android adalah sistem operasi berbasis linux yang dirancang untuk perangkat mobile, seperti telepon pintar (smartphone) dan Komputer Tablet (PDA). Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tanggal 5 November 2007, bersamaan dengan didirikannya Open Handset Alliance (OHA), konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Terdapat dua jenis distributor resmi dari sistem operasi android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Service (GMS) dan yang kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung dari Google atau dikenal sebagai Open Handset Distribution (OHD) [28].

Pada masa saat ini kebanyakan vendor-vendor smartphone sudah memproduksi smartphone berbasis android. Hal ini karena android itu adalah sistem operasi yang open source sehingga bebas didistribusikan dan dipakai oleh vendor manapun. Tidak hanya menjadi sistem operasi di smartphone, saat ini android menjadi pesaing Apple pada sistem operasi Tablet PC. Pesatnya pertumbuhan android adalah karena android itu sendiri merupakan platform yang lengkap, tersedianya tools pengembangan, adanya fasilitas market dan didukung oleh komunitas open source [29].

Dalam perkembangannya, android telah mengalami beberapa pembaruan sejak pertama kali rilis. Android saat tulisan ini dibuat sudah sampai pada generasi ke 12, yang diberi nama Snowcone. Perkembangan versi android dari pertama kali rilis hingga sekarang dapat dilihat pada Tabel 2.4 berikut.

Tabel 2.4 Versi Android

No.	Versi Android	Nama	Tanggal Rilis
1	1.0	Astro/Alpha	23 September 2008
2	1.1	Bender/Beta	9 Februari 2009
3	1.5	Cupcake	30 April 2019
4	1.6	Donut	15 September 2009
5	2.0 – 2.1	Eclair	26 Oktober 2009
6	2.2 – 2.23	Froyo	20 Mei 2010
7	2.3 – 2.37	Gingerbread	6 Desember 2010
8	3.0 – 3.2.6	Honeycomb	22 Februari 2011
9	4.0 – 4.0.4	Ice Cream Sandwich	19 Oktober 2011
10	4.1 – 4.3.1	Jelly Bean	27 Juni 2012
11	4.4 – 4.4.4	Kitkat	31 Oktober 2013
12	5.1 – 5.1.1	Lollipop	12 November 2014
13	6.0 – 6.0.1	Marshmallow	5 Oktober 2015
14	7.1 – 7.1.2	Nougat	22 Agustus 2016
15	8.0 – 8.1	Oreo	21 Agustus 2017
16	9	Pie	6 Agustus 2018
17	10	Quince Tart/ Queen Cake	3 September 2019
18	11	Red Velvet Cake	8 September 2020
19	12	Snow Cone	4 Oktober 2021

2.10.1 Arsitektur Android

Secara umum, arsitektur android terdiri dari lima lapisan, yaitu Applications and Widgets, Applications Framework, Libraries, Android Runtime dan Linux Kernel. Penjelasan dari tiap lapisan tersebut adalah sebagai berikut [29]:

1. Applications and Widgets

Applications dan Widgets merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program, pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam android runtime dengan menggunakan kelas dan service yang tersedia pada framework aplikasi..

2. Applications Framework

Application Framework merupakan lapisan dimana developer melakukan pengembangan/pembangunan aplikasi yang akan dijalankan pada sistem operasi android. Lapisan ini menyediakan berbagai class yang bisa digunakan oleh developer untuk mempermudah dalam proses pengembangan/pembangunan

aplikasi. Termasuk juga berhubungan dengan akses ke hardware dan manajemen user-interface, memori dan sumber daya aplikasi. Komponen yang termasuk Application Frameworks antara lain Views, Content Provider, Resource Manager, Notification, dan Activity Manager.

3. Libraries

Libraries merupakan lapisan dimana fitur-fitur android berada. Developer dapat mengakses libraries untuk menjalankan aplikasinya, lapisan ini berjalan diatas kernel yang didalamnya terdapat kumpulan library C dan C++ seperti Libc dan SSL Selain itu terdapat juga beberapa library lainnya diantaranya:

a. Library Media untuk pemutaran media audio dan video.
b. Library Graphics untuk manajemen tampilan grafis 2D dan 3D.
c. Library SQLite untuk dukungan database.
d. Library LiveWebcore mencakup engine embedded web view.
e. Library SSL dan WebKit untuk integrasi web browser dan keamanan internet.

4. Android Runtime

Android runtime merupakan lapisan yang membuat aplikasi Android dapat dijalankan, dimana dalam prosesnya menggunakan implementasi Linux yang terdiri dari Core Libraries dan Dalvik Virtual Machine (DVM). Core Libraries merupakan serangakaian library Java yang terdiri dari berbagai macam fungsi dasar pada bahasa pemrograman Java, sedangkan DVM merupakan Java Virtual Machine (semacam JVM pada Java ME) yang secara khusus dijalankan pada android.

5. Linux Kernel

Linux kernel merupakan lapisan paling bawah pada arsitektur android, dimana sistem operasi android itu berada. Google menggunakan kernel linux versi 2.6 untuk membangun sistem android, yang mencakup memory management, security setting, power managemnet dan beberapa driver hardware lainnya. Lapisan ini berperan sebagai abstraction layer antara hardware dan keseluruhan software.

2.10.2 Komponen Android

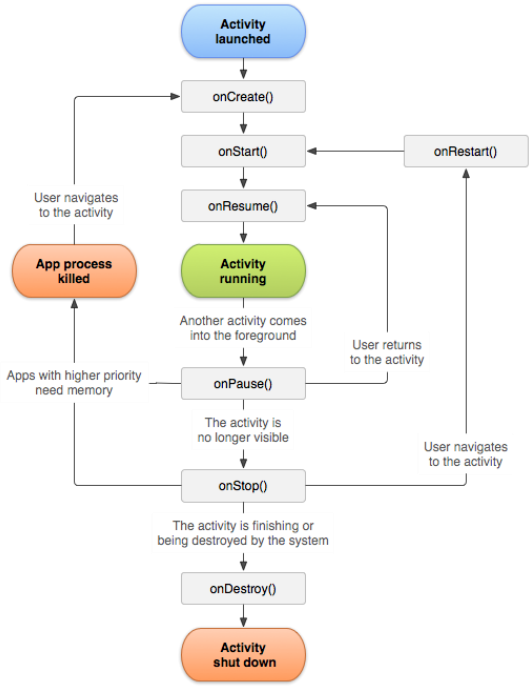
Komponen aplikasi adalah bagian penting dari sebuah aplikasi android. Komponen ini harus terhubung dengan AndroidManifest.xml, file yang

menggambarkan setiap komponen dari aplikasi dan bagaimana mereka berinteraksi. Terdapat enam komponen utama yang dapat digunakan dalam aplikasi android diantranya [29]:

1. Activity

Setiap User Interface diwakili oleh kelas Activity (activity class). Sebuah aplikasi dapat terdiri dari satu atau lebih activity yang diproses dalam Linux, setiap activity mempunyai siklus hidup dimana pada siklus ini untuk menavigasi antara tahap Activity Life-Cycle. Selama siklus ini berjalan, activity bisa mempunyai lebih dari 2 status. Pengguna tidak bisa mengontrol setiap status karena semuanya sudah otomatis ditangani oleh sistem. Namun pengguna akan mendapat pesan saat terjadi perubahan status melalui method callback.

Dalam siklus hidup method callback, cara perilaku activity dapat dideklarasikan saat pengguna meninggalkan dan memasuki kembali ke sebuah activity itu. Disitulah peranan method callback dalam sebuah aplikasi pada platform android. Activity Life-Cycle pada platform Android dapat dilihat pada Gambar 2.6 berikut.



Gambar 2.6 Activity Life-Cycle

Sumber: https://miro.medium.com/max/523/1*BmgNxyQaWUflgZDK96i9cg.png

Dari siklus tersebut dapat diketahui bahwa sebuah kelas Activity menyediakan tujuh method inti callback, Adapun penjelasan dari tiap method callback tersebut yaitu sebagai berikut :

a. onCreate(), Method ini adalah method utama dari setiap Activity. Method ini akan dipanggil saat pertama kali aplikasi dijalankan. Method ini biasanya digunakan pada saat deklarasi variabel atau membuat suatu user interface.
b. onStart(), Method ini dipanggil ketika method onCreate() telah dipanggil. Method ini mengindikasikan activity yang ditampilkan ke pengguna (user).
c. onResume(), Method ini dipanggil ketika method onStart() selesai dipanggil. Method ini adalah keadaan dimana aplikasi mulai berinteraksi dengan pengguna. Aplikasi akan tetap dalam keadaan ini sampai terjadi suatu statement dari aplikasi semisal menerima panggilan telepon atau mematikan layar smartphone.
d. onPause(), Method ini dipanggil ketika pengguna meninggalkan activity (meskipun tidak selalu berarti activity dihancurkan). Method ini berguna untuk menghentikan sementara operasi yang sedang berjalan semisal menjeda pemutaran musik dan lain-lain.
e. onStop(), Method ini dipanggil ketika activity tidak terlihat lagi oleh pengguna, dengan kata lain activity berhenti dijalankan. Hal ini dapat terjadi saat aplikasi berjalan di belakang layar dalam waktu cukup lama. Sistem juga dapat menghubungi method ini ketika activity selesai berjalan, dan akan segera dihentikan.
f. onDestroy(), Method ini dipanggil ketika activity telah selesai dijalankan artinya saat aplikasi benar-benar berhenti, atau karenaproses yang berisi activity tersebut untuk sementara dihancurkan oleh sistem untuk menghemat ruang memori.
g. onRestart(), Method ini dipanggil ketika activity kembali menampilkan user interface setelah status stop.

2. Service

Service digunakan untuk menjalankan proses aplikasi di belakang layar. Sebagai contoh, Service bisa memainkan musik di latar belakang saat pengguna berada dalam aplikasi yang berbeda, atau mungkin mengambil data melalui jaringan tanpa menghalangi interaksi pengguna dengan aktivitas.

3. Content Provider

Content Provider digunakan dalam aplikasi untuk mengakses atau membagikan data (berhubungan dengan database) yang dibutuhkan oleh suatu activity.

4. Resource

Resource digunakan untuk mengatur sumber daya aplikasi dengan cara menyimpan file-file coding dan non-coding yang diperlukan pada sebuah aplikasi kedalam suatu folder khusus. Untuk file coding seperti file XML yang digunakan untuk membentuk sebuah user interface disimpan dalam folder res/layout, sedangkan untuk file non-coding seperti icon disimpan dalam folder res/drawable-ldpi, gambar disimpan dalam folder res/drawable dan audio disimpan dalam folder res/raw.

5. Views

Views digunakan untuk membangun antarmuka pengguna untuk komponen activity yang akan digunakan.

6. Notification

Notification digunakan untuk mekanisme sinyal atau pemberitahuan kepada pengguna secara insidentil sesuai dengan waktu yang telah ditentukan, seperti pesan masuk, telepon masuk dan lain sebagainya yang akan ditampilkan pada status bar.

2.11 Android Studio

Android Studio adalah Integrated Development Environment (IDE) resmi untuk sistem operasi android, pengganti Eclipse Android Development Tools (ADT) sebagai IDE utama untuk pengembangan aplikasi Android. IDE ini diluncurkan pada tanggal 16 Mei 2013. IDE ini dibangun di atas perangkat lunak IntelliJ IDEA JetBrains dan dirancang khusus untuk pengembangan Android. IDE ini tersedia untuk diunduh pada sistem operasi berbasis Windows, macOS dan Linux [30].

Android studio memilki fitur yang sangat membantu developer untuk membangun aplikasi android, berikut ini beberapa fitur dari Android Studio yang di ambil dari situs resmi di https://developer.android.com/studio.

1. Visual Layout Editor

Fitur ini memungkinkan untuk membuat tata letak yang rumit dengan ConstraintLayout dengan menambahkan batasan dari setiap tampilan ke tampilan lainnya. Kemudian pratinjau tata letak dapat disesuaikan pada ukuran layar apa pun dengan memilih salah satu dari berbagai konfigurasi perangkat atau hanya dengan mengubah ukuran jendela pratinjau.

2. APK Analyzer

APK Analyzer memungkinkan developer memeriksa konten file APK, bahkan jika APK tersebut tidak dibuat dengan Android Studio. Seperti memeriksa file manifes, resources, dan file DEX.

3. Emulator

Android studio memilki fitur emulator bawaan sehingga kita dapat langsung melakukan uji coba aplikasi yang telah dibuat langsung dari perangkat komputer tanpa perlu perangkat fisik Android.

4. Flexible Build System

Didukung oleh Gradle, sistem build Android Studio memungkinkan untuk menyesuaikan build untuk menghasilkan beberapa varian build untuk perangkat yang berbeda dari satu proyek Aplikasi Android.

5. Realtime Profilers

Fitur ini menyediakan statistik waktu nyata untuk CPU, memori, dan aktivitas jaringan aplikasi. Identifikasi kinerja dengan merekam jejak metode, memeriksa tumpukan dan alokasi, dan melihat muatan jaringan internet yang masuk dan keluar.

2.12 Java

Sejarah Java dimulai pada tahun 1991. Sebuah tim kecil yang berisikan engineer dari Sun Microsystem mempelopori proyek Java ini. Tim ini disebut Green Team dan dipimpin oleh James Gosling. Saat awal pengembangan, bahasa yang dikembangkan oleh Green Team disebut nama “Greentalk”. Setelah itu, mereka mengganti namanya menjadi “Oak” karena terinspirasi dari sebuah pohon oak yang berada di sebrang kantornya. Pada tahun 1995, Green Team mengganti namanya menjadi “Java” karena Oak sudah menjadi trademark dari Oak

Technologies. Nama “Java” diambil karena terinspirasi dari kopi kesukaan James Gosling yaitu Java Coffe. Pada tahun itu, Sun Microsystem lalu merilis Java untuk pertama kali. Pada 13 November 2006, Sun Microsystem merilis Java dengan gratis dan open source dengan lisensi GNU General Public License (GPL). Tetapi pada tahun 2010, Sun Microsystem dibeli oleh Oracle dan menjadikan Java dikembangkan dan dipelihara di bawah kendali Oracle hingga saat ini [31].

Terdapat tiga platform Java yang masing-masing diarahkan untuk tujuan tertentu dan untuk lingkungan komputasi yang berbeda-beda yaitu:

1. Standard Edition (J2SE), J2SE merupakan inti dari bahasa pemrograman Java. J2SE didesain untuk jalan pada komputer desktop dan komputer workstations.
2. Enterprise Edition (J2EE), Dengan built-in mendukung untuk servlets. JSP, dan XML, edisi ini ditujukan untuk aplikasi berbasis server.
3. Micro Edition (J2ME), Didesain untuk piranti dengan memori terbatas, layar tampilan terbatas dan power pemrosesan yang juga terbatas.

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Java memiliki beberapa kelebihan diantaranya [31]:

1. Simpel, Java merupakan bahasa pemrograman yang simpel, sehingga mudah untuk dipahami. Bermodalkan pengetahuan programming dasar, fundamental dari pemrograman Java akan cepat dipahami.
2. Berorientasi Objek, Java menggunakan konsep pemrograman berorientasi objek, sehingga pembuatan aplikasi bisa dilakukan lebih modular.
3. Kuat, Java didesain untuk membuat aplikasi yang memiliki reliabilitas tinggi. Salah satu cara dari Java untuk membuat program yang memiliki reliabilitas tinggi yaitu dengan mengeliminasi situasi error dengan memeriksanya pada compile time dan runtime.
4. Aman, Java didesain untuk digunakan pada lingkungan yang terdistribusi. Keamanan merupakan hal yang penting dalam lingkungan terdistribusi. Fitur-fitur keamanan yang dimiliki Java membuat perangkat lunak yang dibuat tidak bisa diserang dari luar atau disisipi virus.

5. Arsitektur Netral, Aplikasi yang dibuat menggunakan Java merupakan aplikasi yang platform independent. Aplikasi hanya perlu satu buah versi yang bisa dijalankan pada platform sistem operasi yang berbeda.
6. Portabel, Java dengan karakteristik arsitektur netralnya membuat Java tidak bergantung pada mesin tertentu atau dengan kata lain Java sangatlah portabel.
7. Performa Tingi, Java sangat memperhatikan performa. Dengan pengenalan Just in Time Compilation. Proses kompilasi Java menjadi lebih cepat.
8. Multithreaded, Java memungkinkan pembuatan aplikasi yang bisa melakukan beberapa pekerjaan secara bersamaan.
9. Dinamis, Java lebih dinamis dibandingkan C atau C++ karena desain untuk dijalankan pada lingkungan yang dinamis.

2.13 Kotlin

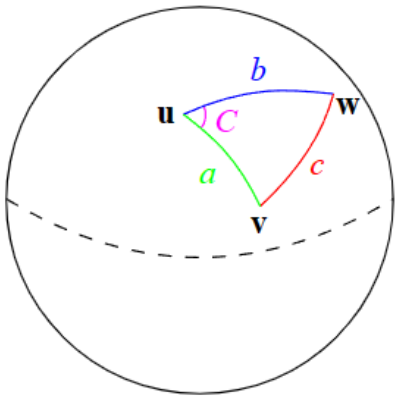
Kotlin adalah bahasa pemrograman statically-typed yang mengkombinasikan prinsip-prinsip object-oriented dengan fitur-fitur fungsional dan berjalan diatas Java Virtual Machine (JVM). Bahasa pemrograman ini dikembangkan oleh JetBrains semenjak 2011 dan resmi didukung oleh Google untuk pengembangan aplikasi android pada Mei 2017 diumumkan pada acara Google I/O 2017. Semenjak event tersebut popularitas bahasa pemrograman ini mengalami peningkatan yang signifikan.

Bahasa pemrograman ini dapat digunakan untuk pengembangan aplikasi android, server-side dan client-side. Kotlin dipersiapkan full interoperability dan dengan Java, sehingga Kotlin dapat digabungkan dalam satu project aplikasi dengan bahasa Java. Masalah lain yang biasa ditemukan dalambahasa pemrograman Java adalah NullExceptionPointer (NPE).

Kotlin didesain null-safety, sehingga masalah tersebut tidak lagi ditemui dalam Kotlin. Pada bahasa Kotlin, NPE sudah dapat diketahui pada saat compile time, berbeda dengan Java yang melakukan pengecekan NPE pada saat runtime. Kotlin juga didukung dengan fitur functional programming, seperti penggunaan lambda expression, higher-order function, lazy evaluation dan beberapa method pada collections seperti filtering, mapping, ordering dan lain-lain.

2.14 Algoritma Haversine Formula

Algorima haversine formula merupakan sebuah metode yang digunakan dalam sistem navigasi dimana metode ini akan menghasilkan sebuah perhitungan jarak antara dua titik dari garis bujur (longitude) dan garis lintang (latitude) [1]. Haversine formula merupakan suatu cara penentuan jarak dari titik koordinat berdasarkan posisi garis lintang dan garis bujur atau dalam aplikasinya kini menggunakan latitude dan longitude pada google maps, hasil dari perhitungan dengan metode haversine formula adalah jarak dari kedua titik yang dapat digambarkan dalam peta menggunakan fasilitas API pada google maps. Bentuk pola haversine formula dapat dilihat pada Gambar 2.7 berikut.



Gambar 2.7 Bentuk Pola Haversine Formula

Sumber: https://blogs.itb.ac.id/anugraha/2014/09/10/teori-pengukuran-jarak/

Pada gambar 1 merupakan gambaran dari pola haversine formula yang digambarkan dalam bentuk trigonometri bola yang mana persamaan ini adalah persamaan yang amat penting dalam sistem navigasi, nantinya haversine formula ini akan menghasilkan jarak terpendek antara dua titik. Formula ini awalnya digunakan untuk masalah utama astronomi nautical yaitu untuk menentukan jarak antar bintang. Digunakan pertama kali oleh Josef de Mendoza y Rios di tahun 1801, dan formula ini ditemukan oleh Jamez Andrew di tahun 1805. Istilah harvesine sendiri diciptakan atau dinamakan pada tahun 1835 oleh Prof. James Inman.

Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.3671 km, dan lokasi dari 2 titik di koordinant bola (lintang dan bujur)

masing-masing adalah lonh1, lat1, dan lonh2, lat2, maka rumus haversine formula dapat ditulis dengan persamaan berikut:

$$x = (long2 - long1) * \cos(\frac{lat1+lat2}{2}) \text{ …... (1)}$$

$$y = (lat2 - lat1) \text{ ……. (2)}$$

$$d = sqrt(x * x + y * y) * R \text{ ……..... (3)}$$

Ket : lat1 = derajat latitude pasien

lat2 = derajat latitude faskes

long1 = derajat longitude pasien

long2 = derajat longitude faskes

x = selisih longitude (bujur)

y = selisih latitude (lintang)

d = jarak (km)

R = jari-jari bumi (6371 km)

1 derajat = 0,0174532925 radian

2.15 Algoritma Fuzzy Logic

Logika Fuzzy pertama kali dikembangkan oleh Lotfi A. Zadeh pada tahun 1965. Teori ini banyak diterapkan di berbagai bidang, antara lain representasi pikiran manusia kedalam suatu sistem. Banyak alasan mengapa penggunaan logika fuzzy ini sering dipergunakan antara lain, konsep logika fuzzy yang mirip dengan konsep berpikir manusia. Sistem fuzzy dapat merepresentasikan pengetahuan manusia ke dalam bentuk matematis dengan lebih menyerupai cara berpikir manusia.

Pengontrol dengan logika fuzzy mempunyai kelebihan yaitu dapat mengontrol sistem yang kompleks, non-linier, atau sistem yang sulit direpresentasikan kedalam bentuk matematis. Selain itu, informasi berupa pengetahuan dan pengalaman mempunyai peranan penting dalam mengenali perilaku sistem di dunia nyata. Logika fuzzy juga memiliki himpunan fuzzy yang mana pada dasarnya, teori himpunan fuzzy merupakan perluasan dari teori himpunan klasik. Dimana dengan logika fuzzy, hasil yang keluar tidak akan selalu konstan dengan input yang ada.

Cara kerja logika fuzzy secara garis besar terdiri dari input, proses dan output. Logika fuzzy merupakan suatu teori himpunan logika yang dikembangkan untuk mengatasi konsep nilai yang terdapat diantara kebenaran (truth) dan kesalahan (false). Dengan menggunakan fuzzy logic nilai yang dihasilkan bukan hanya ya (1) atau tidak (0) tetapi seluruh kemungkinan diantara 0 dan 1.

2.16 Firebase

Firebase adalah API yang disediakan google untuk penyimpanan dan penyelarasan data ke dalam aplikasi Android, iOS, atau web. Firebase ini merupakan solusi yang ditawarkan oleh Google untuk mempermudah pekerjaan Mobile Apps Developer. Dengan adanya Firebase, developer bisa fokus mengembangkan aplikasi tanpa harus memberikan effort yang besar untuk urusan backend [32]. Firebase dibangun diatas tiga pilar yang Develop, Grow, dan Earn. Untuk lengkapnya dapat dilihat pada Gambar 2.8 berikut

Gambar 2.8 Pilar Firebase

Sumber: https://storage.googleapis.com/gweb-uniblog-publish-prod/images/How_Using_Firebase_Can_Help_You_Earn_More.max-1000x1000.png

Salah satu fitur yang dimiliki oleh Firebase yaitu Firebase Real Time Database, fitur ini yang memberikan sebuah NoSQL database yang bisa diakses secara real time oleh pengguna aplikasi. Dan hebatnya adalah aplikasi bisa menyimpan data

secara lokal ketika tidak ada akses internet, kemudian melakukan sync data segera setelah mendapatkan akses internet [32].

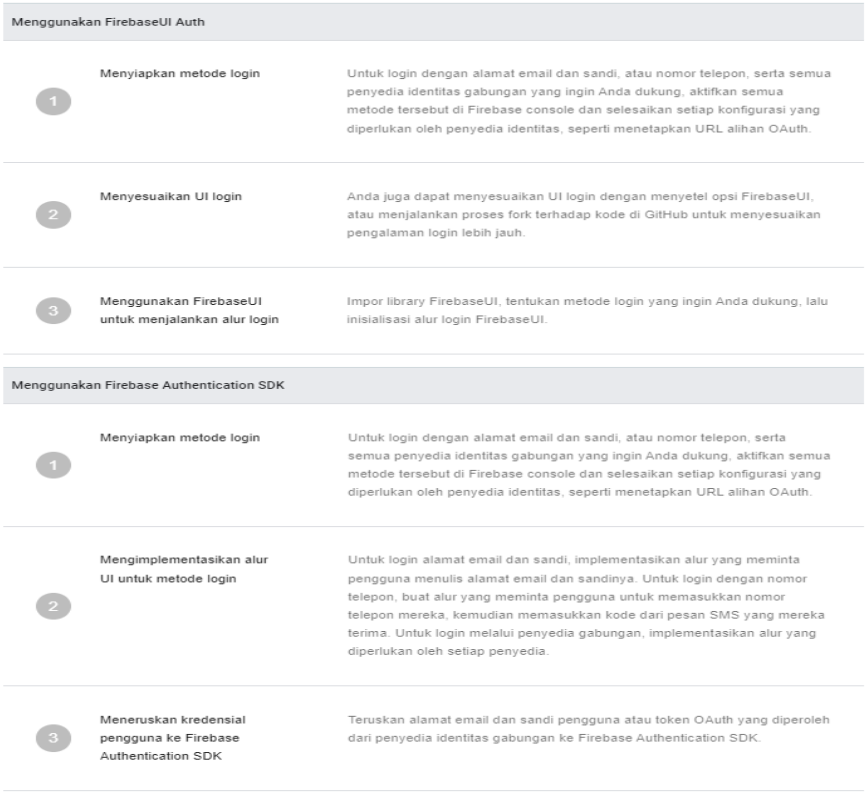
Tetapi fitur pada firebase bukan hanya realtime database, jauh lebih dari itu. Firebase memiliki banyak fitur seperti authentication, database, storage, hosting, pemberitahuan dan lain-lain. Pada penelitian ini fitur Firebase yang digunakan adalah Firebase Authentication, Firebase Realtime Database dan Firebase Cloud Messaging.

2.16.1 Firebase Authentication

Firebase Authentication merupakan layanan siap pakai yang dimiliki oleh Firebase SDK. Firebase Authentaction memungkinkan aplikasi untuk melakukan autentikasi yang aman, sekaligus meningkatkan pengalaman login dan pengalaman aktivasi bagi end-user. Metode autentikasi yang digunakan meliputi email and password-based authentication, federated identity provider identegration (authentication menggunakan akun Google, Facebook, Twitter atau Github), custom authentication system integration hingga anonymous authentication.

Firebase User Authentication ini bekerja dengan cara mengirimkan server response dari Firebase Server berdasarkan credential yang dikirimkan oleh client ke Firebase Server. Credential tersebut dapat berupa alamat email dan password ataupun sebuah token OAuth dari sebuah federated identity provider. Melalui server response yang diterima dari Firebase Server, aplikasi dapat mengakses informasi dasar profil pengguna dan mengontrol akses pengguna terhadap produk atau layanan Firebase yang terdapat pada aplikasi.

Firebase Authentication dibangun untuk memberikan API yang mudah kepada pengembang yang dapat digunakan untuk proses sign-in dari federated providers dengan skema email dan password, atau yang sudah terintegrasi dengan autentikasi yang sudah ada. Firebase Authentication telah terintegrasi dengan Firebase Realtime Database sehingga administrator dapat mengontrol siapa yang dapat mengakses data. Alur implementasi dari Firebase Authentication dapat dilihat pada Gambar 2.9 berikut.



Gambar 2.9 Alur Implementasi Firebase Authentication

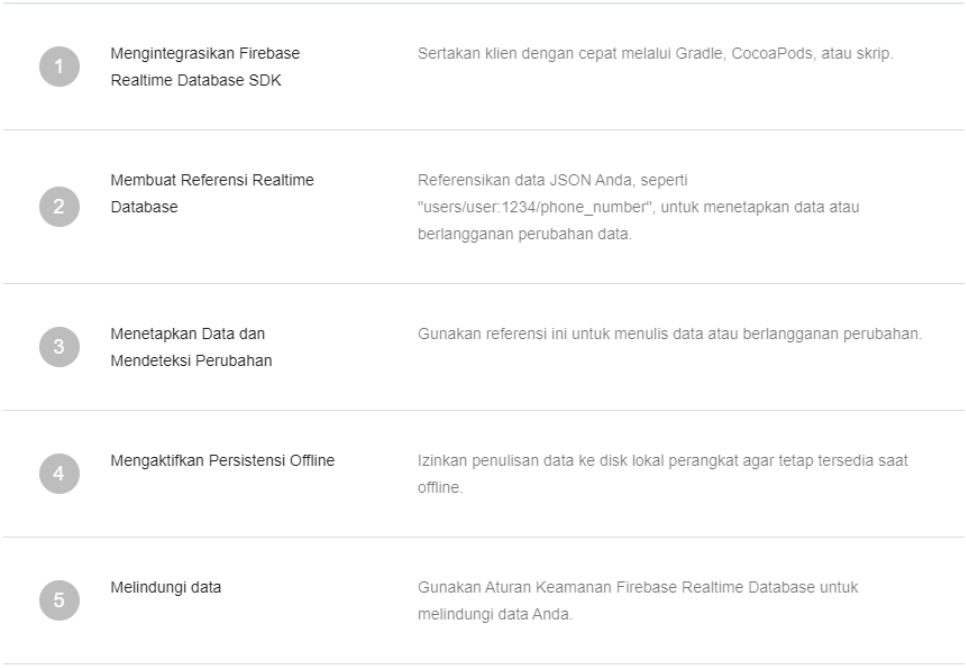
Sumber: https://firebase.google.com/docs/auth?hl=id

2.16.2 Firebase Realtime Database

Firebase Realtime Database adalah database NoSQL yang di-host di cloud. Data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung. Ketika devoloper membuat aplikasi lintas-platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah instance Realtime Database dan menerima update data terbaru secara otomatis [33].

Firebase Realtime Database memungkinkan devoloper untuk membuat aplikasi kolaboratif dan kaya fitur dengan menyediakan akses yang aman ke database, langsung dari kode sisi klien. Data disimpan di drive lokal. Bahkan saat offline sekalipun, peristiwa realtime terus berlangsung, sehingga pengguna akhir akan merasakan pengalaman yang responsif. Ketika koneksi perangkat pulih kembali, Realtime Database akan menyinkronkan perubahan data lokal dengan update jarak jauh yang terjadi selama klien offline, sehingga setiap perbedaan akan otomatis digabungkan [33].

Firebase Realtime Database menyediakan bahasa aturan berbasis ekspresi yang fleksibel, atau disebut juga Aturan Keamanan Firebase Realtime Database, untuk menentukan metode strukturisasi data dan kapan data dapat dibaca atau ditulis. Ketika diintegrasikan dengan Firebase Authentication, developer dapat menentukan siapa yang memiliki akses ke data tertentu dan bagaimana mereka dapat mengaksesnya [33]. Alur implementasi dari Firebase Realtime Database dapat dilihat pada Gambar 2.10 berikut.



Gambar 2.10 Alur Implementasi Firebase Realtime Database

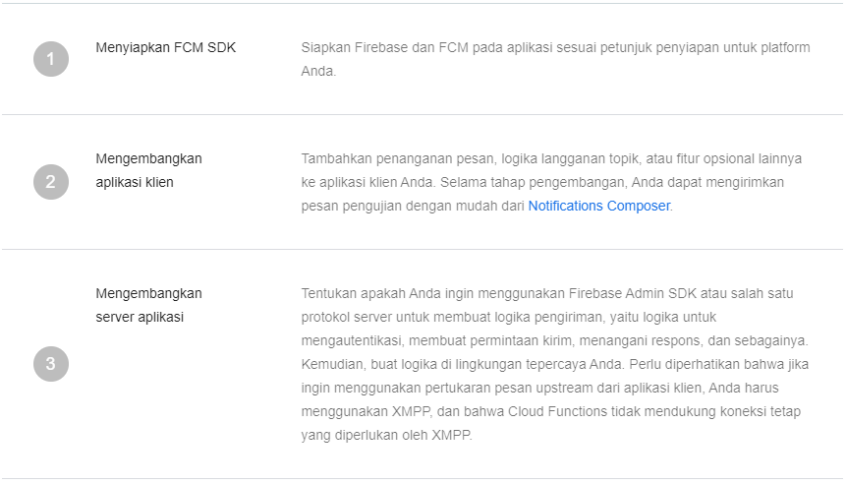
Sumber: https://firebase.google.com/docs/ database/?hl=id

2.16.3 Firebase Cloud Mesagging

Firebase Cloud Messaging (FCM) adalah solusi pengiriman pesan lintas platform untuk mengirim pesan dan layanan pemberitahuan yang disediakan oleh Google secara gratis.

Layanan FCM juga menyediakan fungsi untuk melakukan push notification, yaitu notifikasi yang muncul dibagian atas layar smartphone dan dapat diseret ke bawah, untuk mengakses pesan lengkapnya pengguna cukup menekan pesan yang tampil pada notifikasinya. Penggunaan Fitur push notification dengan FCM sangat membantu karena FCM akan mengirimkan notifikasi secara realtime [34].

Fitur-fitur yang diberikan oleh FCM sebenarnya tidak terlalu jauh berbeda dengan Google Cloud Messaging. Dengan FCM kita bisa memberikan pembertihauan dan membuat komunikasi dua arah antara perangkat [34]. Alur implementasi dari FCM dapat dilihat pada Gambar 2.11 berikut.

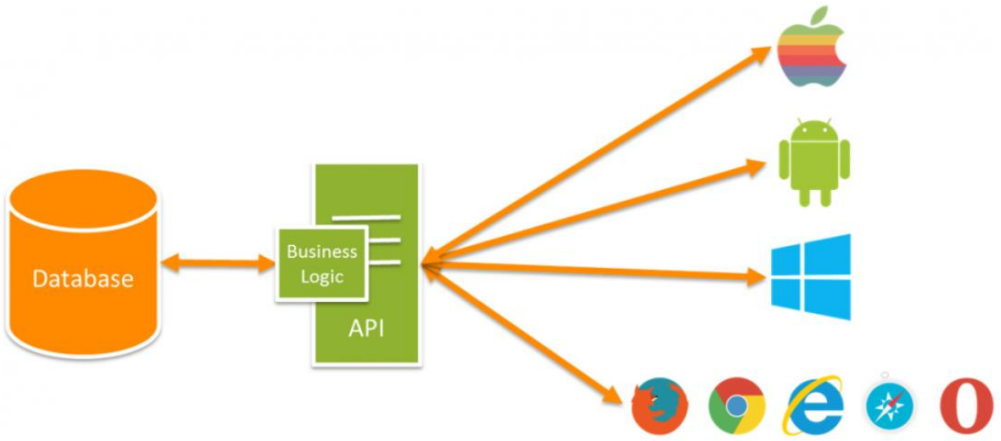


Gambar 2.11 Alur Implementasi Firebase Cloud Messaging
Sumber: https://firebase.google.com/docs/cloud-messaging/?hl=id

2.17 Application Programming Interface (API)

Application Programming Interface (API) adalah software interface yang memiliki intruksi dan disimpan dalam bentuk library dan dapat terhubung atau berinteraksi dengan software lain. Secara struktural, API merupakan spesifikasi dari suatu struktur data, object, function, beserta parameter-parameter yang diperlukan untuk mengakses resource dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu interface yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain [35].

Tujuan penggunaan API adalah untuk mempercepat proses development dengan menyediakan function secara terpisah sehingga developer tidak perlu membuat fitur yang serupa. Fungsi dari API ini pada umumnya adalah sebagai sumber data yang bisa digunakan untuk kebutuhan sistem atau aplikasi tertentu, API memungkinkan untuk dapat bertukar informasi data dari satu aplikasi ke aplikasi lainnya. Gambaran umum cara kerja API dapat dilihat pada Gambar 2.12 berikut.



Gambar 2.12 Cara Kerja API

Sumber: https://www.codepolitan.com/mengenal-apa-itu-web-api-5a0c2855799c8

Selain sebagai sumber data API juga mempercepat proses development dengan menyediakan fungsi secara terpisah sehingga developer tidak perlu membuat fungsi yang sama. API sendiri terdiri dari beberapa elemen seperti functions, protocols, dan tools untuk membantu developer membuat aplikasi.

Untuk melakukan request kepada API client harus memenuhi beberapa syarat tertentu sebagai berikut [35]:

1. URL

Dalam HTTP (Hyper Text Transfer Protocol) URL adalah alamat unik dari suatu web atau sistem tertentu dan akan di arahkan sesuai proses bisnis yang ada pada server. URL dapat dibuat berupa halaman sebuah website, gambar, video, atau data tertentu.

2. Method

Request method yang di lakukan oleh client meberitahu server bahwa aksi apa yang perlu di lakukan untuk memenuhi permintaan client. Ada 4 method yang biasanya digunakan yaitu :

a. GET – meminta server untuk memberikan respon sumber.
b. POST – meminta server untuk membuat sumber baru.
c. PUT – meminta server untuk melakukan edit/update pada sumber yang sudah ada.
d. DELETE – meminta server untuk melakukan penghapusan sumber.

3. Headers

Headers biasanya berisi meta-informasi seperti waktu permintaan, format data yang dinginkan, atau informasi perangkat yang digunakan oleh client sehingga server dapat megirimkan data sesuai dengan perangkat yang digunakan.

4. Body

Body berisi data yang ingin dikirmkan client ke server, sehingga server dapat melakukan respon aksi berdasarkan data yang dikirimkan client ke server.

2.17.1 Google Maps API

Google Maps API adalah salah satu Application Programming Interface (API) yang dimiliki Google. API ini mempunyai fitur untuk melakukan aktivitas-aktivias yang berkaitan dengan Google Maps, antara lain menampilkan peta, mencari rute terdekat antara dua tempat, dan lain sebagainya. Google Maps API tersedia untuk platform android, iOS, web, dan juga web service. Google Maps API juga menyediakan layanan seperti direction, geocoding, distance matrix API, dan elevation API.

Google Maps merupakan sebuah layanan gratis yang diberikan oleh google yang dapat kita gunakan untuk dapat untuk dapat melihat suatu daerah. Kita dapat menambahkan fitur google maps dalam sistem atau aplikasi yang kita buat. Google Maps API adalah suatu library yang berbentuk javascript dimana kita dapat mengubah dan menambahkan variabel-variabel tertentu sehingga bisa dibuat sesuai dengan keinginan kita. Berkas yang mengandung bytecode kemudian akan dikonversikan oleh java interpreter menjadi bahasa mesin sesuai dengan jenis dan platform yang digunakan [36].

Dalam perkembangannya Google Maps API diberikan kemampuan untuk mengambil peta statis, melakukan geocoding dan memberikan penuntun arah. Kekurangan pada Google Maps API yaitu jika ingin melakukan akses harus terdapat layanan internet pada perangkat yang digunakan, sedangkan kelebihan yang dimiliki yaitu:

1. Dukungan penuh yang dilakukan google sehingga terjamin dan fitur yang bervariasi pada google maps API.

2. Banyak pengembang yang menggunakan Google Maps API sehingga mudah dalam mencari referensi dalam pengembangan aplikasi.

2.17.2 Google Places API

Google Places API adalah antarmuka / interface yang disediakan oleh Google untuk para pengembang yang ingin mendapatkan data tentang tempat - tempat yang terdaftar di Google Maps.

Apabila kita membuka Google Maps kita dapat melihat semacam marker-marker penanda tempat, yang jika diklik akan muncul data tentang tempat tersebut, seperti alamat, nama tempat, koordinat latitude, longitude, dan sebagainya, itulah data yang dimaksud.

Dengan adanya Google Places API ini, pengembang dapat mengambil data tersebut untuk digunakan pada aplikasi yang dibuat. Ada berbagai macam fitur yang tersedia pada Google Places API seperti place picker, get your place, autocomplete, dan sebagainya.

2.17.3 Covid-19 Indonesia API

Covid-19 Indonesia API adalah antarmuka / interface public (open sources) yang bisa digunakan oleh para pengembang untuk mendapatkan data statistik kasus covid-19 di Indonesia. Adapun data yang disediakan berasal dari website resmi https://data.covid19.go.id/.

2.17.4 RS Bed Covid Indonesia API

RS Bed Covid-19 Indonesia API adalah antarmuka / interface public (open sources) yang bisa digunakan oleh para pengembang untuk mendapatkan data ketersediaan rumah sakit dan tempat tidur rumah sakit untuk pasien covid-19 ataupun non-covid yang berada di Indonesia. Adapun data yang disediakan berasal dari website resmi https://data.covid19.go.id/.

2.17.5 Indonesia News API

Indonesia News API adalah antarmuka / interface public (open sources) yang bisa digunakan oleh para pengembang untuk mendapatkan data artikel berita dari berbagai portal berita di Indonesia.

2.18 Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa pemrograman berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C [37].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. UML merupakan bahasa pemodelan yang bertujuan untuk pengembangan dalam bidang rekayasa perangkat lunak yang dimaksudkan untuk memberikan cara standar untuk memvisualisasikan desain suatu sistem. Penciptaan UML pada awalnya dimotivasi oleh keinginan untuk membakukan sistem notasi yang berbeda dan pendekatan untuk desain perangkat lunak [38].

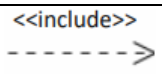
Berdasarkan pemaparan mengenai UML dari beberapa sumber referensi, maka dapat disimpulkan UML merupakan alat bantu dalam melakukan pemodelan yang saling berhubungan secara langsung dalam pembangunan sebuah sistem agar lebih efektif. Di dalam UML terdapat berbagai macam diagram, empat di antaranya yang akan digunakan pada penelitian ini adalah Use Case Diagram, Activity Diagram, Sequence Diagram, dan Class Diagram.

2.18.1 Use Case Diagram

Use Case Diagram adalah uraian sekelompok yang saling terkait satu dengan lainnya dan membentuk sistem secara teratur yang dilakukan oleh sebuah aktor.

Use case digunakan untuk membentuk tingkah laku suatu benda dalam sebuah model serta direalisasikan oleh sebuah kolaborasi. Hal yang ditekankan dalam diagram ini adalah “apa” yang dapat diperbuat oleh sistem bukan “bagaimana”. Use case merepresentasikan interaksi antara user dengan sistem dan menyatakan sebuah aktivitas atas pekerjaan tertentu. Aktor merepresentasikan sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan–pekerjaan tertentu [39]. Simbol penggunaan use case diagram ditunjukan pada Tabel 2.5 berikut [40], [41].

Tabel 2.5 Simbol Use Case Diagram

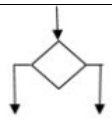
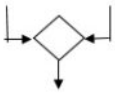
No.	Simbol	Nama	Keterangan
1		Actor	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
2		System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
3		Use Case	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
4		Collaboration	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
5		Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi
6		Depedency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent).
7		Generalization	Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya.
8		Association	Komunikasi antar aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
9		Include	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan

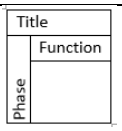
No.	Simbol	Nama	Keterangan
			memerlukan use case ini untuk menjalankan fungsinya
10	<<extend>> ------>	Extend	Relasi use case tambahan ke sebuah use case, dimana use case yang ditambahkan dapat berdiri sendiri.

2.18.2 Activity Diagram

Activity Diagram menggambarkan *workflow* proses bisnis dan urutan aktivitas dalam sebuah proses. Diagram ini mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat activity diagram pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. Activity diagram juga bermanfaat untuk menggambarkan *parallel behaviour* atau menggambarkan interaksi antara beberapa usecase [39]. Simbol-simbol yang ada pada activity diagram ditunjukan pada Tabel 2.6 berikut [40], [41].

Tabel 2.6 Simbol Activity Diagram

No.	Simbol	Nama	Deskripsi
1		Initial Code	Menandakan awal dimulainya suatu proses
2		Actions	Menggambarkan urutan dari proses atau akitivitas.
3		Flow	Menunjukkan berjalannya suatu proses.
4		Decision	Memberikan dua pilihan kondisi dengan sebuah arah *flow* masuk dan dua atau lebih arah *flow* yang keluar.
5		Merge	Menyatukan *flow* yang sebelumnya terpisah dari proses *decision*.
6		Fork	Ketika sebuah *flow* masuk dan terdapat dua *flow* keluar yang menjalankan dua aktivitas bersamaan.
7		Join	ketika terdapat dua *flow* masuk yang menjalankan dua aktivitas bersamaan dan hanya terdapat sebuah *flow* yang keluar.
8		Activity Final	Menunjukkan akhir dari suatu proses

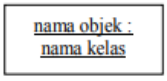
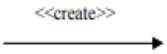
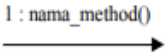
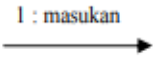
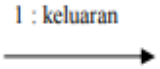
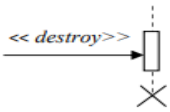
No.	Simbol	Nama	Deskripsi
9		Subactivity Indicator	Menandakan suatu proses atau aktivitas yang dipecah pada activity diagram lain. Hal ini bertujuan agar activity diagram tidak terlalu kompleks
10		Connector	Mempermudah ketika activity diagram sudah terlalu kompleks. Sebuah flow masuk ke suatu connector yang telah diberi nama dan flow keluar dari connector dengan nama sama yang menunjuk suatu decision atau actions.
11		Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.18.3 Sequence Diagram

Sequence Diagram menunjukkan interaksi objek yang diatur dalam urutan waktu. Ini menggambarkan objek dan kelas yang terlibat dalam skenario dan urutan pesan yang dipertukarkan antara objek yang diperlukan untuk menjalankan fungsionalitas skenario. Sequence diagram biasanya dikaitkan dengan realisasi use case dalam logical view dari sistem yang sedang dikembangkan. Sequence diagram kadang-kadang disebut diagram acara atau skenario acara. Sebuah sequence diagram sebagai garis-garis paralel paralel menunjukkan berbagai proses atau objek yang hidup secara bersamaan, dan, sebagai panah horizontal, pesan-pesan dipertukarkan di antara mereka, dalam urutan terjadinya. Ini memungkinkan spesifikasi skenario runtime sederhana secara grafis [38]. Simbol-simbol yang ada pada sequence diagram ditunjukan pada Tabel 2.7 berikut [40], [41].

Tabel 2.7 Simbol Sequence Diagram

No.	Simbol	Nama	Keterangan
1		Aktor	Orang, poses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang

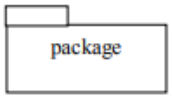
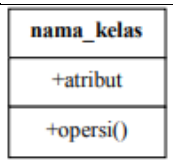
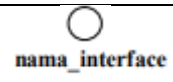
No.	Simbol	Nama	Keterangan
2		Garis hidup / Lifeline	Menyatakan kehidupan suatu objek.
3		Objek	Menyatakan objek yang berinteraksi pesan
4		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
5		Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6		Pesan tipe call	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
7		Pesan tipe send	Menyatakan bahwa sutu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
8		Pesan tipe return	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
9		Pesan tipe destroy	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy
10		Pesan kembali	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.
11		Alternatif	Mengambil keputusan/tindakan untuk suatu kondisi tertentu.

2.18.4 Class Diagram

Dalam rekayasa perangkat lunak, class diagram dalam UML adalah jenis diagram struktur statis yang menggambarkan struktur sistem dengan menunjukkan kelas sistem, atributnya, operasi (atau metode), dan hubungan antar objek. Class diagram adalah blok bangunan utama pemodelan berorientasi objek. Ini digunakan untuk pemodelan konseptual umum dari struktur aplikasi, dan untuk pemodelan terperinci menerjemahkan model ke dalam kode pemrograman. Class diagram juga dapat digunakan untuk pemodelan data. Kelas-kelas dalam class diagram mewakili

elemen utama, interaksi dalam aplikasi, dan kelas yang akan diprogram [38]. Simbol-simbol yang ada pada class diagram ditunjukan pada Tabel 2.8 berikut [40], [41].

Tabel 2.8 Simbol Class Diagram

No.	Simbol	Nama	Keterangan
1		Package	Package merupakan sebuah bungkusan dari satu atau lebih kelas
2		Kelas	Kelas pada struktur sistem
3		Antarmuka / interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
4		Asosiasi / association	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
5		Asosiasi berarah	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
6		Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
7		Kebergantungan / depedency	Relasi antar kelas dengan makna kebergantungan antar kelas
8		Agregasi / aggregation	Relasi antar kelas dengan makna semua-bagian (whole-part)