BAB 2
LANDASAN TEORI

2.1 Jatuh

Jatuh merupakan suatu kejadian yang diberitahukan penderita atau saksi mata, yang melihat kejadian yang mengakibatkan seseorang mendadak terbaring/terduduk di lantai/tempat yang lebih rendah dengan atau tanpa kehilangan kesadaran atau luka [1].

Banyak faktor yang memengaruhi kejadian jatuh, baik faktor seperti kelemahan otot ekstrimitas bawah, kekakuan sendi, *sinkop* dan pusing, serta faktor ekstrinsik seperti lantai licin, hilangnya keseimbangan, dan menurunnya daya penglihatan.

2.1.1 Komplikasi Jatuh

Jatuh dapat mengakibatkan berbagai jenis cedera, kerusakan fisik dan psikologis. Kerusakan fisik yang paling diwaspadai dari kejadian jatuh adalah patah tulang panggul. Jenis fraktur lain yang sering terjadi akibat kejadian jatuh adalah fraktur pergelangan tangan, lengan atas dan *pelvis* serta kerusakan pada jaringan lunak. Komplikasi-komplikasi jatuh yaitu :

a. Perlukaan (*Injury*)

   Jatuh dapat mengakibatkan berbagai jenis cedera, kerusakan fisik dan psikologis. Kerusakan fisik yang paling ditakuti dari kejadian jatuh adalah patah tulang panggul. Jenis fraktur lain yang sering terjadi akibat jatuh adalah fraktur pergelangan tangan, lengan atas dan pelvis serta kerusakan jaringan lunak.

b. Disabilitas

   Mengakibatkan penurunan mobilitas yang berhubungan dengan perlukaan fisik dan penurunan mobilitas akibat jatuh yaitu kehilangan kepercayaan diri dan pembatasan gerak.

c. Kerusakan saraf tulang belakang atau tulang ekor.

d. Patah tulang.
e. Cedera kepala.
f. Pendarahan internal.
g. *Solusio* plasenta (lepasnya plasenta dari dinding rahim bagian dalam sebelum proses persalinan).
h. Pecahnya uterus dan membran.
i. Kematian.

2.1.2 Faktor Resiko

Untuk dapat memahami faktor resiko jatuh, maka harus dimengerti bahwa stabilitas badan ditentukan atau dibentuk oleh:

1. Sistem Sensorik

   Yang berperan di dalamnya adalah: visus, pendengaran, fungsi *vestibuler*, dan *proprioseptif*. *Vertigo* tipe *perifer* sering terjadi pada lanjut usia, diduga karena perubahan fungsi vestibuler akibat proses menua. Neuropati *perifer* dan penyakit *degeneratif* leher dapat menganggu fungsi *proprioseptif*.
2. Sistem Saraf Pusat

   SSP akan memberikan respon motorik untuk mengantisipasi input sensorik. Penyakit SSP seperti stroke, *parkinson*, *hidrosefalus* dengan tekanan normal, yang diderita oleh lanjut usia akan menyebabkan gangguan fungsi SSP sehingga berespon tidak baik terhadap input sensorik.
3. Kognitif

   Pada beberapa penelitian, demensia diasosiasikan dengan meningkatnya risiko jatuh.
4. Muskuloskeletal

   Gangguan muskuloskeletal menyebabkan gangguan gaya berjalan (gait) dan ini berhubungan dengan proses menua yang fisiologis. Gangguan gait yang terjadi akibat proses menua tersebut antara lain di sebabkan oleh:

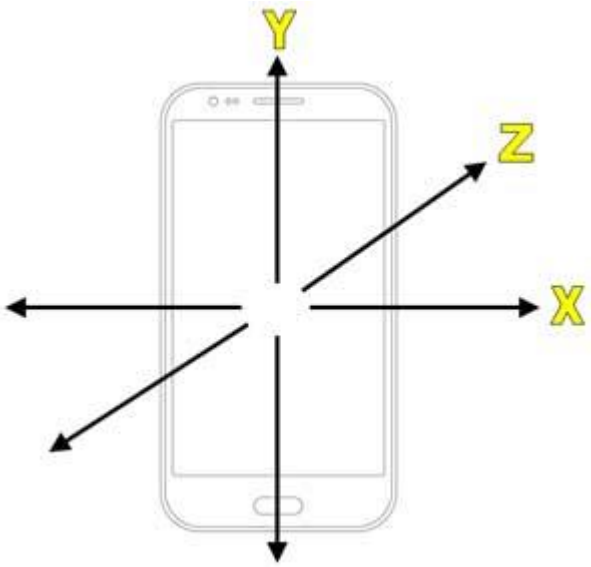
   a. Kekakuan jaringan penghubung.
   b. Berkurangnya massa otot.
   c. Perlambatan konduksi saraf.

d. Penurunan *visus.*
e. Kerusakan *proprioseptif*.

2.2 Sensor *Accelerometer*

Sensor adalah sebuah piranti yang dapat mengubah suatu besaran fisis, misal : jarak, cahaya, panas, magnet, listrik, dan lain sebagainya, ke bentuk besaran fisis lainnya.

Sensor *accelerometer* merupakan perangkat elektronik yang digunakan untuk mengukur percepatan dari suatu objek [10]. *Accelerometer* ini mengukur percepatan, bahwa perangkat mengalami perubahan yang sesuai dengan nilai dari tiga sumbu X,Y,Z lalu data dari tiga sumbu tersebut akan dihitung untuk mengetahui percepatan atau perlambatan yang terjadi. Gambar sumbu *accelerometer* dapat dilihat pada Gambar 2.1 Sumbu *Accelerometer*.



Gambar 2.1 Sumbu *Accelerometer*

Sumber : https://mainthebest.com

Sensor *accelerometer* dapat digunakan untuk kebutuhan seperti mengubah tampilan layar dari portrait ke landscape atau sebaliknya dengan memiringkan bodi ponsel, ini terjadi karena ada perubahan nilai x, y, z dari *smartphone* [11]. nilai sumbu *accelerometer* dapat dilihat pada Tabel 2.1. Nilai Sumbu *Accelerometer.*

Tabel 2.1 Nilai Sumbu *Accelerometer*

Position	x	y	z
vertical	0	1	0
vertical upside down	0	-1	0
right landscape	1	0	0
left landscape	-1	0	0
flat	0	0	1

Percepatan adalah suatu keadaan yang mengakibatkan perubahan bertambahnya kecepatan terhadap waktu, sedangkan apabila perubahan berkurangnya kecepatan terhadap waktu disebut perlambatan. *Accelerometer* yang diletakkan di permukaan bumi dapat mendeteksi percepatan 1g (ukuran gravitasi bumi) pada titik vertikalnya. Sedangkan percepatan yang arah pergerakannya secara horizontal, sensor *accelerometer* akan mengukur percepatannya secara langsung ketika bergerak secara horizontal [12].

Prinsip dari *accelerometer* adalah sebuah per dengan beban dan dilepaskan, beban bergerak dengan suatu percepatan sampai kondisi tertentu lalu berhenti. Bila ada sesuatu yang mengguncangkannya maka beban berayun kembali. *Accelerometer* dapat digunakan untuk mengukur getaran pada mobil, mesin, bangunan, dan instalasi pengamanan. Sensor *accelerometer* juga dapat diaplikasikan pada pengukuran aktivitas gempa, pedometer atau penghitung langkah.

2.3 *Heart Rate Variability*

Heart rate variability adalah fenomena fisiologis dimana terjadi variasi interval waktu antar detak jantung [13]. HRV juga umumnya dikenal dengan istilah R-R interval karena umumnya yang diukur adalah interval waktu pada sinyal EKG (elektrokardiogram). Hal-hal yang dapat mempengaruhi antara lain pengaruh pengaruh sistem saraf otonom, volume darah yang kembali ke jantung (*venous terurn*), respirasi, penyakit aritmia, dan lain sebagainya. Pada tahun 1996, perkumpulan masyarakat eropa para dokter ahli jantung atau *Task Forces of The*

European Society of Cardiology and The North American Society of Pacing and Electrophysiologi yang melibatkan para ahli kesehatan, teknik matematika dan fisiologis mengeluarkan suatu pedoman standar pengukuran, interpretasi fisiologis, dan penggunaan klinis (*standard of measurement, physicological interpretation, and clinical use*) untuk analisa sinyal EKG (elektrokardiogram) yang dinamakan HRV (*heart rate variability*). Dengan mengunakan metoda HRV, seseorang dapat diketahui perubahan aktivitas jantungnya dan dianalisa untuk menginterpertasikan keadaan jantung. Analisa EKG (elektrokardiogram) dengan metoda HRV berfokus terhadap perubahan osilasi interval waktu detak jantung yang berurutan juga kecepatan detak jantung. Oleh sebab itu HRV atau heart rate variability digunakan untuk menggambarkan variasi interval RR dan kecepatan detak jantung.

Normalnya detak jantung akan berdetak secara teratur, 60-100 kali per menit. Apabila diatas 100 kali permenit disebut takikardia, Gejala ini biasanya dapat terjadi pada saat orang yang tekanan intrakarnial meningkat, dan efek samping beberapa obat. Tekanan intrakarnial adalah mengalami peningkatan tekanan otak normal [14]. Sedangkan apabila di bawah 60 kali per menit disebut bradikardia.

2.3.1 Bradikardia

Bradikardia adalah kondisi dimana detak jantung berdetak lebih lambat dari biasanya. Hal ini dapat diartikan bahwa jantung tidak bekerja dengan baik, sehingga detak jantung menjadi sangat lambat dan tidak dapat memompa darah untuk memenuhi kebutuhan tubuh. Terlebih lagi itu adalah salah satu organ penting dalam tubuh manusia. Oleh karena itu pengecekan denyut jantung sangatlah penting. Detak jantung yang melambat tidak menimbulkan gejala. Namun apabila sering terjadi dan disertai dengan aritmia, detak jantung yang lambat dapat menimbulkan gangguan pada organ dan jaringan tubuh lain yang pasokan darahnya tidak terpenuhi. Ketika pasokan darah ke organ atau jaringan tubuh terganggu, gejala yang akan muncul adalah pusing, pingsan, lemas, sakit kepala, nyeri dada, gangguan penglihatan, dan sakit kepala [15]. gejala tersebut memiliki resiko yang berbahaya.

Menurut dr. Inez Leonita, bradikardia lebih sering dialami oleh lansia hal ini dapat disebabkan karena gangguan anatomis, gangguan elektroslit, dan gangguan jantung. Pada ibu hamil sendiri yang cenderung terjadi adalah takikardia karena pembuluh darah ibu kini harus memperdarahi bayinya juga sehingga kerja jantung dan pembuluh darah semakin berat. Normalnya ibu hamil sendiri mengalami peningkatan volume darah 40 sampai 50 persen agar bisa memberikan nutrisi yang cukup pada bayi sehingga menyebabkan detak jantung meningkat. Bradikardia pada ibu hamil juga dapat disebabkan oleh gangguan otot jantung, dan infeksi jantung.

Berikut gambar tabel panduan detak jantung pada Gambar 2.2 Tabel Panduan Target Detak Jantung.

Age	Target HR Zone 50-85%	Average Maximum Heart Rate, 100%
20 years	100-170 beats per minute	200 beats per minute
30 years	95-162 beats per minute	190 beats per minute
35 years	93-157 beats per minute	185 beats per minute
40 years	90-153 beats per minute	180 beats per minute
45 years	88-149 beats per minute	175 beats per minute
50 years	85-145 beats per minute	170 beats per minute
55 years	83-140 beats per minute	165 beats per minute
60 years	80-136 beats per minute	160 beats per minute
65 years	78-132 beats per minute	155 beats per minute
70 years	75-128 beats per minute	150 beats per minute

Gambar 2.2 Tabel Panduan Target Detak Jantung

Sumber : https://hellosehat.com

Bradikardia yang parah dan tidak mendapat penanganan dapat menimbulkan komplikasi berupa :

a. *Sinkop* (pingsan).
b. Pusing.
c. Hipertensi.

d. Gagal jantung.
e. Hipotensi.

2.4 *Wearable Device*

Wearable device adalah sebuah teknologi elektronika atau komputer yang disematkan pada sebuah benda yang dapat dikenakan di tubuh [16]. *Wearable computer* pertama kali ditemukan oleh Ed Thorp dan Claude Shannon pada tahun 1966 berupa komputer analog seukuran kotak rokok yang digunakan untuk menebak roda *roullette*. Setahun setelahnya, Hobert Upton membuat *wearable analog computer* yang digunakan di mata untuk membantu membaca bahasa bibir.

Pada tahun 2013 fitbit mengeluarkan *fitness tracker* (*Fitbit Surge*) berupa *smartwatch* yang kompatibel dengan sistem operasi Android dan iOS. Smartwatch adalah slah satu bagian dari banyaknya varian *wearable device* [17]. Varian lain dari *wearable device* adalah smart *eyeglass* untuk *augmented reality* dan virtual *relaity*, *wristband* dan *ankleband* untuk *health* dan *fitness*, dan *wearable device* yang menangkap motion.

2.4.1 Mi Band 2

Mi Band 2 merupakan perangkat *wearable device* yang termasuk dalam varian *wristband*, ada beberapa fitur yang dimiliki oleh perangkat ini yaitu [18]:

1. *Water Resistant* dengan spesifikasi IP67.
2. *Chipset bluetooth* 4.0.
3. Memiliki sensor detak jantung.
4. *Sleep tracking* yang dapat mendeteksi kualitas tidur penggunanya.
5. Sport Tracker yang dapat memperlihatkan berapa langkah yang telah dilampaui oleh pengguna.

2.4.2 *Bluetooth*

Bluetooth merupakan teknologi yang berkembang sebagai jawaban atas kebutuhan komunikasi antar perlengkapan elektronik agar dapat saling mempertukarkan data dalam jarak yang terbatas menggunakan gelombang radio

dengan frekuensi tertentu. *Bluetooth* beroperasi dalam pita frekuensi 2,4 Ghz dengan menggunakan sebuah *frequency hopping traceiver* yang mampu menyediakan layanan komunikasi data dan suara secara real time antara host-host Bluetooth dengan jarak terbatas. Kelemahan teknologi ini adalah jangkauannya yang pendek dan kemampuan transfer data yang rendah.

Pada dasarnya *Bluetooth* diciptakan untuk menggantikan atau menghilangkan penggunaan kabel didalam pertukaran informasi, dengan konsumsi daya yang rendah, interoperability yang menjanjikan, mudah dalam pengoperasian dan mampu menyediakan layanan yang bermacam-macam.

Bluetooth 4.0 adalah teknologi *Bluetooth* versi terbaru dengan keunggulan utama hemat energi yang resmi diadopsi oleh *Bluetooth Special Interest Group* (SIG) pada tahun 2010. Teknologi nirkabel baru ini dapat digunakan di berbagai perangkat dengan konsumsi energi yang rendah, berbeda dengan spesifikasi bluetooth sebelumnya. Penerapan perangkat *Bluetooth* 4.0 akan memungkinkan perbaikan yang antara lain berupa tingkat penggunaan energi minimum, rata-rata konsumsi daya modus idle (statis), dan kemampuan untuk menjalankan fungsinya selama bertahun-tahun pada standar baterai coin-cell [19]. *Bluetooth* 4.0 memiliki beberapa keunggulan yaitu :

1. Konsumsi daya yang lebih kecil sehingga hemat baterai atau listrik.
2. Waktu pemakaian yang lebih lama sampai tahunan dengan hanya menggunakan baterai kapasitas kecil (*coin cell batteries*).
3. Biaya produksi yang rendah.
4. Jarak yang lebih panjang. Mampu menjangkau lebih dari 100 meter.
5. Kecepatan transfer yang lebih baik dengan kemampuan sekitar 1 Mbps.
6. Sistem keamanan yang lebih terkontrok dibandingkan dengan sistem keamanan pada *Bluetooth* generasi-generasi sebelumnya.

2.5 *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 e*levation API.*

Google maps merupakan sebuah layanan gratis yang diberikan oleh *google* dan sangat populer. *Google maps* adalah suatu peta dunia yang dapat kita gunakan untuk dapat untuk dapat melihat suatu daerah. Dengan kata lain, *google maps* merupakan suatu peta yang dapat dilihat dengan menggunakan suatu *web browser*. 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 [20].

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.

2.6 *Location Based Service*

Sistem layan berbasis lokasi atau yang biasa dikenal *dengan location based services* (LBS), menggabungkan antara proses dari layanan *mobile* dengan posisi geografis dari penggunanya. LBS mampu mendeteksi lokasi pengguna berada sehingga dapat meberikan layanan sesuai dengan lokasi pengguna tersebut. Lokasi geografis pengguna ditentukan dengan menggunakan layanan terpisah seperti misalnya *Global Positioning System* (GPS). yang mana GPS adalah sistem yang

berfungsi sebagai sistem navigasi global yang dapat menerima informasi dari sistem satelit. Satelit GPS ini memancarkan sinyal yang memungkinkan penerima sinyal GPS untuk mendapatkan informasi berupa posisi, kecepatan, maupun waktu [21].

Tingkat ketelitian posisi GPS bervariasi dari sangat teliti (kesalahan beberapa mm) sampai kurang teliti (kesalahan puluhan meter). Tingkat ketelitian ini ditentukan oleh tingkat ketelitian data satelit, geometri satelit, metode penetuan posisi dan pengolahan data, serta kecanggihan alat penerima.

Berdasarkan posisi dari pengguna, LBS memungkinkan untuk menemukan lokasi-lokasi penting seperti restoran, toko, hotel, rumah sakit, tempat wisata, menghitung rute, atau mendapatkan informasi lainnya.

Deida (2007) menyebutkan *Location Based Services* (LBS) memiliki setidaknya lima komponen utama dalam arsitekturnya yaitu:

1. *Mobile Device*

   Merupakan komponen penting dalam LBS, *mobile device* ini digunakan untuk meminta informasi yang diinginkan oleh pengguna, hasil yang diterima dapat berupa gambar, suara, teks dan sebagainya. *mobile device* yang digunakan dapt berupa *smartphone* atau PDA.

2. *Communication Network*

   Digunakan untuk menyampaikan informasi *query* dan lokasi *mobile device* ke penyedia layanan dan mengirim hasil dari penyedia layanan ke *mobile device.* Jaringan yang mungkin dapat digunakan antara lain *wireless wide area network* (WWAN) seperti GSM dan UMTS, *wireless area local network* (WLAN) seperti IEEE 802.11 atau personal *network* seperti *bluetooth*.

3. *Positioning Component*

   Digunakan untuk memberikan informasi lokasi pengguna, posisi pengguna dapat diperoleh melalui jaringan komunikasi ponsel (*celltriangulasi*) atau dapat juga menggunakan sebuah penerima GPS.

4. *Service Application Provider*

   Penyedia layanan merupakan komponen LBS yang memberikan berbagai macam layanan yang bisa digunakan oleh pengguna. Sebagai contoh ketika

pengguna meminta layanan agar bisa tahu posisinya saat itu, maka aplikasi dan penyedia layanan langsung memproses permintaan tersebut, mulai dari menghitung dan menentukan posisi pengguna, menemukan rute jalan, mencari data di *yellow pages* sesuai dengan permintaan dan masih banyak lainnya.

5. *Data Provider*

   Digunakan untuk menyimpan data mengenai layanan yang dapat diberikan melalui *location based services* seperti informasi lokasi, restoran, rumah sakit, pom bensin, tempat wisata, dan lain sebagainya.

2.7 Android

Menurut Safaat, Nazruddin (2012) Android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis pada sistem operasi Linux [22]. Android bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode - kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler.

Setiap aplikasi memiliki tingkatan yang sama. Android tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. API yang disediakan menawarkan akses ke hardware, maupun data-data ponsel sekalipun, atau data sistem itu sendiri. Bahkan pengguna dapat menghapus aplikasi inti dan menggantikannya dengan aplikasi pihak ketiga. Sedangkan Android SDK (*Software Development Kit*) menyediakan Tools dan API yang diperlukan untuk mengembangkan aplikasi pada platform Android dengan menggunakan bahasa pemrograman Java. Adapun kelebihan dan kelemahan android [23]:

A. Kelebihan Android

Android merupakan salah satu *operating system* terbanyak yang dipakai pada *smartphone* karena beberapa kelebihan sistem operasi android seperti di bawah ini:

1. *Open Source :* Android merupakan turunan linux untuk sistem operasi pada telepon selular yang otomatis menjadikan android OS berbasis open source. Hal ini membuat banyak pengembang aplikasi yang membuatnya untuk sistem operasi android ini.
2. Android semakin berkembang dari waktu ke waktu: ini dibuktikan dengan perkembangan versi android pada perangkat-perangkat *smartphone* yang kini merambah pula pada komputer tablet.
3. Akses market aplikasi yang mudah: Android memiliki playstore untuk mengunduh langsung aplikasi-aplikasi apa saja yang diinginkan dan banyak pula aplikasi gratis yang dapat diakses.
4. Dukungan *google*: menggunakan android seperti memiliki *google* ditangan anda karena sangat *support* terhadap layanan *google* seperti *google maps*, gmail, *google* reader dan masih banyak lagi.
5. *Custom* ROM: banyak aplikasi custom Rom untuk android selain sangat menyenangkan karena dapat dengan bebas memodifikasi interface penggunaannya aplikasi aman untuk perangkat.

B. Kelemahan Android

Banyaknya kelebihan sistem operasi android tidak menutup celah kekurangan yang ada. Beberapa diantaranya adalah :

1. Ponsel android boros baterai: Memang tidak bisa dipungkiri *smartphone* lebih boros baterai dari pada telepon selular biasa namun, OS android memang lebih boros karena banyak proses OS berjalan di *background* sehingga baterai cepat habis.
2. *Playstore* kurang terkontrol: aplikasi untuk android dapat diunggah secara mudah pada *playstore* namun tidak menutup kemungkinan aplikasi-aplikasi *malware* menyusup menyerupai aplikasi.

2.8 Java

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan java tidak hanya terfokus pada satu sistem operasi, tetapi juga dikembangkan untuk berbagai sistem operasi dan bersifat *open source* [24]. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana. Aplikasi-aplikasi berbasis Java
umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi Java mampu berjalan di beberapa platform sistem operasi yang berbeda, Java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Java menggunakan model pengamanan tiga lapis (*three-layer security model*) untuk melindungi sistem dari *untrusted* Java code. Pertama, bytecode *verifier* membaca bytecode sebelum dijalankan dan menjamin bytecode memenuhi aturan-aturan dasar bahasa Java. Kedua, *class loader* menangani pemuatan kelas Java ke *runtime interpreter*. Ketiga, manajer keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem file, port jaringan, proses eksternal dan sistem window.

Java termasuk bahasa *Multithreading*. *Thread* adalah untuk menyatakan program komputer melakukan lebih dari satu tugas di satu waktu yang sama. Java menyediakan kelas untuk menulis program *multithreaded*, program mempunyai lebih dari satu thread eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren.

Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek-objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di bahasa yang memungkinkan alokasi dinamis.

Java mempunyai mekanisme *exception-handling* yang ampuh. *Exceptionhandling* menyediakan cara untuk memisahkan antara bagian penanganan kesalahan dengan bagian kode normal sehingga menuntun ke struktur kode program yang lebih bersih dan menjadikan aplikasi lebih tegar. Ketika kesalahan yang serius ditemukan, program Java menciptakan *exception. Exception* dapat ditangkap dan dikelola program tanpa resiko membuat sistem menjadi turun.

Program Java mendukung native method yaitu fungsi ditulis di bahasa lain, biasanya C/C++. Dukungan native method memungkinkan pemrogram menulis fungsi yang dapat dieksekusi lebih cepat dibanding fungsi ekivalen di java. Native *method* secara dinamis akan di-*link* ke program java, yaitu diasosiasikan dengan program saat berjalan.

Adapun kelebihan dari Java yaitu :

Java mempunyai beberapa keunggulan dibandingkan dengan Bahasa pemrograman lainnya. Keunggulan bahasa pemrograman Java antara lain:

1. Berorientasi objek

   Java adalah bahasa pemrograman yang berorientasi pada objek. Java membagi program menjadi objek - objek serta memodelkan sifat dan tingkah laku masing-masing dalam menyelesaikan suatu masalah.

2. Java bersifat *multiplatform*

   Java dirancang untuk mendukung aplikasi yang dapat beroperasi di lingkungan jaringan berbeda. Untuk mengakomodasi hal tersebut, Java compiler membangkitkan bytecodes (sebuah format yang tidak tergantung pada arsitektur tertentu yang didesain untuk mengirimkan kode ke banyak platform perangkat keras dan perangkat lunak secara efisien). Java dapat dijalankan oleh banyak platform seperti Linux, Unix, Windows, Solari, maupun Mac.

3. Java bersifat *multithread*

   Multithreading adalah kemampuan suatu program komputer untuk mengerjakan beberapa proses dalam suatu waktu. Thread dalam Java memiliki kemampuan untuk memanfaatkan kelebihan multi prosessor apabila sistem operasi yang digunakan mendukung multi prosessor.

4. Dapat didistribusi dengan mudah

   Java memiliki *library* rutin yang lengkap untuk dirangkai pada *protocol* TCP/IP (seperti HTTP dan FTP) dengan mudah. Kemampuan networking Java lebih kuat dan lebih mudah digunakan. Java memudahkan tugas pemrograman jaringan yang sulit seperti membuka dan mengakses sebuah soket koneksi. Java juga memudahkan pembuatan CGI (*Common Gateway Interface*).

5. Bersifat dinamis

   Java dirancang untuk beradaptasi dengan lingkungan yang sedang berkembang. Java bersifat dinamis dalam tahap *linking*. Class yang ada dapat di-*link* sebatas yang diperlukan, apabila diperlukan modul kode yang baru dapat di-*link* dari beberapa sumber, bahkan dari sumber dalam jaringan Internet.

Kekurangan Java yaitu:

1. Tulis sekali

   Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.

2. Mudah didekompilasi

   Dekompilasi adalah proses membalikkan dari kode jadi menjadi kod sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET *Platform*. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-*reverse-engineer*.

3. Penggunaan memori yang banyak

Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena *trend* memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.9 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu – Integrated Development Environment (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA [25]. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas pengembang saat membuat aplikasi Android, misalnya:

1. Sistem versi berbasis Gradle yang fleksibel.
2. Emulator yang cepat dan kaya fitur.
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
5. *Template* kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat pengujian dan kerangka kerja yang ekstensif.
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk *Google* Cloud Platform, mempermudah pengintegrasian *Google* Cloud Messaging dan App Engine.

Tujuan dibuatnya Android Studio adalah untuk mempercepat pengembangan dan membantu pengembang membuat aplikasi berkualitas tinggi untuk setiap perangkat Android.

Menawarkan alat bantu yang dibuat khusus untuk pengembang Android, meliputi pengeditan kode yang lengkap, *debugging*, pengujian, dan alat pembuatan profil.

2.10 *Firebase*

Firebase merupakan sebuah layanan infrastruktur *Backend-as-a-service* (BaaS) yang diakuisisi oleh *Google* pada oktober 2014. *Firebase* menawarkan kemudahan kepada para pengembang perangkat lunak dalam membangun aplikasi yang lebih baik serta bisnis yang sukses melalui seluruh fitur komplementernya. *Firebase* dibangun diatas tiga pilar yang meliputi *Develop, Grow, dan Earn*, dapat dilihat pada Gambar 2.3 Pilar *Firebase* [26].

Gambar 2.3 Pilar *Firebase*

Sumber : *The Definitive Guide to Firebase*, 2017

Firebase memiliki fitur yang dirancang untuk meningkatkan pengalaman pengembang perangkat lunak dalam mengembangkan aplikasinya. Berikut adalah beberapa teknologi tersebut :

2.10.1 *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.

Untuk membuat pengguna *login* ke aplikasi, dapatkan kredensial autentikasi dari pengguna terlebih dahulu. Kredensial ini dapat berupa alamat email dan sandi pengguna atau token OAuth dari penyedia identitas tergabung. Kemudian, teruskan kredensial ini ke *Firebase Authentication* SDK. Layanan backend kami selanjutnya akan memverifikasi kredensial tersebut dan menampilkan respons ke klien.
Setelah berhasil *login*, Anda dapat mengakses informasi profil dasar pengguna dan Anda dapat mengontrol akses pengguna ke data yang disimpan di produk *Firebase* lainnya. Anda juga dapat menggunakan token autentikasi yang disediakan untuk memverifikasi identitas pengguna di layanan backend Anda sendiri.

2.10.2 *Firebase Realtime Database*

Firebase Realtime Database merupakan sebuah layanan NoSQL *cloudhosted database* yang dimiliki oleh *Firebase* SDK. *Firebase Realtime Database* adalah *database event-driver* yang cara kerjanya berbeda dari *database* SQL. Tidak ada kode sisi *server* dan tingkat akses *database* semua pengkodean dilakukan di klien. Layanan ini menawarkan layanan penyimpanan data yang dapat

disinkronisasikan secara *realtime* terhadap seluruh klien yang terhubung. Maksud dari *realtime* adalah jika terdapat perubahan pada data pada *database*, maka seluruh *client* yang terhubung akan secara otomatis mendapatkan perubahannya dalam hitungan milidetik. Kemudian *offline*, yaitu aplikasi yang menggunakan *Firebase Realtime Database* akan tetap responsif bahkan saat *offline*. Hal ini disebabkan karena *Firebase* SDK dapat mempertahankan data dan perubahannya pada media penyimpanan klien. Pada saat klien terhubung ke jaringan internet, maka *Firebase* SDK akan melakukan penyesuaian otomatis atas catatan perubahan data yang disimpan pada media penyimpanan klien dengan kondisi terkini dari *Firebase* server. Kemampuan inti yang terakhir adalah *accessible* from *client devices*. Layanan ini menawarkan kemudahan untuk mengakses *Firebase Realtime Database* secara langsung dari sebuah perangkat mobile atau sebuah peramban web tanpa membutuhkan *server application*.

Firebase Realtime Database memungkinkan Anda 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.

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.

Realtime Database adalah *database* NoSQL, sehingga memiliki pengoptimalan dan fungsionalitas yang berbeda dengan *database* terkait. API Realtime *Database* dirancang agar hanya mengizinkan operasi yang dapat dijalankan dengan cepat. Hal ini memungkinkan Anda untuk membangun

pengalaman *realtime* yang luar biasa dan dapat melayani jutaan pengguna tanpa mengorbankan kemampuan respons. Oleh karena itu, perlu dipikirkan bagaimana pengguna mengakses data, kemudian buat struktur data sesuai dengan kebutuhan tersebut.

2.10.3 *Firebase Cloud Storage*

Firebase Cloud Storage dirancang untuk pengembang perangkat lunak yang perlu menyimpan dan menampilkan konten buatan pengguna seperti foto, audio, video ataupun file lainnya. *Firebase Cloud Storage* memberikan kemudahan dengan memberikan API yang didukung oleh *Google Cloud Storage*. *Firebase* Cloud Storage SDK mendapat dukungan dari *Google security* untuk mengunggah dan mengunduh file, termasuk apa yang harus dilakukan untuk menangani kegagalan komunikasi dan melanjutkan apa yang tertinggal.

Pengembang menggunakan SDK *Firebase* untuk *Cloud Storage* untuk mengunggah dan mengunduh file langsung dari klien. Jika koneksi jaringan buruk, klien dapat mencoba kembali operasi tepat di mana ia tinggalkan, menghemat waktu dan *bandwidth* pengguna Anda.

Cloud Storage menyimpan file Anda di *Google Cloud Storage Bucket*, membuatnya dapat diakses melalui *Firebase* dan *Google Cloud*. Ini memungkinkan fleksibilitas untuk mengunggah dan mengunduh file dari klien seluler melalui *Firebase* SDK, dan melakukan pemrosesan sisi server seperti pemfilteran gambar atau video *transcoding* menggunakan *Google Cloud Platform*. *Cloud Storage* berskala otomatis, artinya tidak perlu bermigrasi ke penyedia lain. Pelajari lebih lanjut tentang semua manfaat integrasi kami dengan *Google Cloud Platform*.

Firebase SDKs *for Cloud Storage* terintegrasi secara mulus dengan *Firebase Authentication* untuk mengidentifikasi pengguna, dan menyediakan bahasa keamanan deklaratif yang memungkinkan untuk mengatur kontrol akses pada file individual atau grup file, sehingga dapat menjadikan file sebagai publik atau pribadi sesuai keinginan.

Developer menggunakan *Firebase* SDK untuk *Cloud Storage* untuk mengunggah dan mengunduh file langsung dari klien. Jika koneksi jaringan buruk,

klien bisa mencoba operasi ini lagi dari posisi terakhir saat pengoperasian terhenti, sehingga menghemat waktu dan *bandwidth* pengguna.

Cloud Storage menyimpan file Anda di bucket *Google Cloud Storage*, sehingga membuatnya mudah diakses melalui *Firebase* dan *Google Cloud*. Dengan begitu, Anda memiliki fleksibilitas untuk mengunggah dan mengunduh file dari klien seluler melalui *Firebase* SDK, dan melakukan pemrosesan sisi server seperti pem-filteran gambar atau transcoding video menggunakan *Google Cloud Platform*. *Cloud Storage* akan diskalakan secara otomatis, yang berarti tidak perlu bermigrasi ke penyedia lain. Pelajari lebih lanjut tentang semua manfaat integrasi dengan *Google Cloud Platform* kami.

Firebase SDK untuk *Cloud Storage* terintegrasi sempurna dengan *Firebase Authentication* untuk mengidentifikasi pengguna dan kami menyediakan bahasa keamanan deklaratif yang dapat Anda gunakan untuk menyetel kontrol akses pada masing-masing file atau kumpulan file, sehingga Anda bisa menyetelnya sebagai publik atau pribadi sesuai keinginan.

2.10.4 *Firebase Cloud Messaging*

Firebase Cloud Messaging adalah solusi messaging lintas *platform* yang memungkinkan terjadinya komunikasi dua arah antara perangkat. Dengan menggunakan FCM, pengembang dapat memberitahu aplikasi klien bahwa email baru atau data lain tersedia untuk disinkronkan. Pengembang dapat mengirim pesan pemberitahuan untuk mendorong keterlibatan dan retensi keterlibatan pengguna. Dengan menggunakan FCM memungkinkan terjadinya pengiriman dua jenis pesan ke klien :

1. *Notification messages*, terkadang diangg *Display messages* ditangani oleh FCM SDK secara otomatis. Digunakan FCM pengembang ingin menangani pemberitahuan pada aplikasi klien.
2. Data *messages*, yang ditangani oleh aplikasi klien. Digunakan jika pengembang ingin memproses pesan di aplikasi klien.

Pesan pemberitahuan berisi serangkaian kunci yang dapat dilihat pengguna yang telah ditentukan sebelumnya. Pesan data, sebaliknya, hanya berisi pasangan

nilai kunci khusus yang ditetapkan pengguna. Pesan pemberitahuan dapat berisi muatan data opsional. *Payload* maksimum untuk kedua jenis pesan adalah 4KB, kecuali saat mengirim pesan dari *Firebase console*, yang memberlakukan batas 1024 karakter.

FCM dapat mengirim pesan pemberitahuan termasuk muatan data opsional. Dalam kasus semacam itu, FCM menangani tampilan payload notifikasi, dan aplikasi klien menangani payload data.

Implementasi FCM mencakup dua komponen utama untuk mengirim dan menerima:

1. Lingkungan terpercaya seperti *Cloud Functions* untuk *Firebase* atau server aplikasi yang digunakan untuk membuat, menargetkan, dan mengirim pesan. Aplikasi klien iOS, Android, atau web (*JavaScript*) yang menerima pesan.
2. Anda dapat mengirim pesan melalui SDK Admin atau API HTTP dan XMPP. Untuk menguji atau mengirim pesan pemasaran atau keterlibatan dengan penargetan dan analitik internal yang kuat, Anda juga dapat menggunakan komposer *Notifications*.

2.11 Pengertian dan Konsep dari Model *Waterfall*

Metode *Waterfall* adalah sebuah metode pengembangan sistem dimana antar satu fase ke fase yang lain dilakukan secara berurutan. Dalam proses implementasi metode *Waterfall* ini, sebuah langkah akan diselesaikan terlebih dahulu dimulai dari tahapan yang pertama sebelum melanjutkan ke tahapan yang berikutnya. Adapun keuntungan menggunakan metode *waterfall* ini yaitu *requirement* harus didefinisikan lebih mendalam sebelum proses *coding* dilakukan, selain itu proses implementasinya dilakukan secara bertahap dari tahap pertama hingga tahap terakhir secara berurutan. Disamping itu metode *Waterfall* ini juga memungkinkan sedikit mungkin perubahan yang dilakukan oleh proyek berlangsung.

Adapun metode *Waterfall* menurut Ian Sommerville [9], metode *waterfall* memiliki tahapan utama dari waterfall model yang mencerminkan aktifitas

pengembangan dasar. Terdapat 5 (lima) tahapan pada metode *Waterfall*, yaitu *Requirements Definition, System and Software Design, Implementation and Unit Testing, Integration and System Testing, Operation and Maintenance.*

Adapun penjelasan dari tahapan-tahapan metode *waterfall* menurut Ian Sommerville tersebut sebagai berikut.

1. *Requirement Analysis and Definition* adalah tahapan penetapan fitur, kendala dan tujuan sistem melalui konsultasi dengan pengguna sistem. Semua hal tersebut akan ditetapkan secara rinci dan berfungsi sebagai spesifikasi sistem.
2. Pada Tahap *System and Software Design* ini akan dibentuk suatu arsitektur sistem berdasarkan persyaratan yang telah ditetapkan. Sekain itu juga, dilakukan identifikasi dan penggambaran terhadap abstraksi dasar sistem perangkat lunak beserta hubungan-hubungannya.
3. Dalam tahapan *Implementation and Unit Testing* ini, hasil dari desain perangkat lunak akan direalisasikan sebagai satu set program atau unit program. Setiap unit akan diuji apakah sudah memenuhi spesifikasinya.
4. Dalam tahap *Integration and System* Testing ini, setiap unit program akan diintegrasikan satu sama lain dan diuji sebagai satu sistem yang utuh untuk memastikan sistem sudah memenuhi persyaratan yang ada. Setelah itu sistem akan dikirim ke pengguna sistem.
5. Dalam tahap *Operation and Maintenance* ini, sistem diinstal dan mulai digunakan. Selain itu juga memperbaiki error yang tidak ditemukan pada tahap pembuatan. Dalam tahap ini juga dilakukan pengembangan sistem seperti penambahan fitur dan fungsi baru.

2.12 *Unified Modelling Language*

Unified Modelling Language (UML) adalah sebuah bahasa pemodelan *visual* yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan artefak sistem perangkat lunak [27]. UML digunakan untuk memahami , mendesain, mengeksplorasi, mengkonfigurasi, memelihara, dan mengontrol informasi tentang sistem. Hal Ini dimaksudkan untuk digunakan

dengan semua metode pengembangan, tahapan siklus hidup, domain aplikasi, dan media.

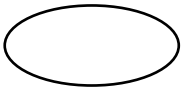
UML bukanlah bahasa pemrograman, melainkan alat yang dapat menyediakan generator kode dari UML ke dalam berbagai bahasa pemrograman serta membangun model rekayasa balik dari program yang ada. UML adalah bahasa pemodelan diskrit, yang dimaksudkan untuk menjadi bahasa pemodelan umum untuk sistem diskrit seperti yang terbuat dari perangkat lunak, *firmware*, atau logika digital.

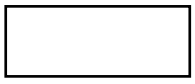
UML diadopsi dengan suara bulat oleh keanggotaan *Object Management Group* (OMG) sebagai standar pada November 1997. OMG memiliki tanggung jawab untuk pengembangan lebih lanjut dari standar UML. Bahkan sebelum adopsi terakhir, sejumlah buku diterbitkan yang menguraikan hal-hal penting dari UML.

2.13 *Usecase* Diagram

Use case diagram dapat digunakan selama proses analisis untuk menangkap requirements sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *usecase* diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan . Dalam sebuah model mungkin terdapat satu atau beberapa *usecase* diagram. Tabel simbol *usecase* diagram dapat dilihat pada Tabel 2.2. Simbol *Usecase* Diagram.

Tabel 2.2 Simbol *Usecase* Diagram

No.	Simbol	Nama	Keterangan
1		*Actor*	Merupakan orang, proses atau *system* lain yang berinteraksi dengan *system* yang akan dibuat. Jadi walaupun simbol aktor dalam diagram *usecase* berbentuk orang, namun aktor belum tentu orang.
2		*Use case*	merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling berinteraksi atau

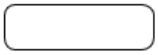
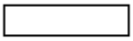
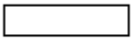
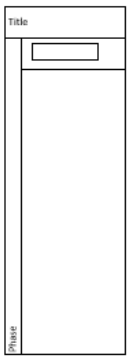
			bertukar pesan antar unit maupun aktor.
3		*Association*	Merupakan relasi yang digunakan untuk menggambarkan interaksi antara *usecase* dan aktor. Asosiasi juga menggambarkan berapa banyak objek lain yang bisa berinteraksi dengan suatu objek atau disebut *multiplicity* .
4		*Extend*	Menghubungkan antara satu objek dengan objek lain.
5		*Generalization*	Hubungan dimana objek berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk.
6		*Include*	Menspesifikasi bahwa *usecase* sumber secara eksplisit.
7		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
8		*System Boundary*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

2.14 *Activity* Diagram

Activity diagram memodelkan alur kerja (*work flow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah flowchart karena user dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (*state*). Diagram Activity berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu

proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Tabel simbol *activity* diagram dapat dilihat pada Tabel 2.3 Simbol *Activity* Diagram.

Tabel 2.3 Simbol *Activity* Diagram

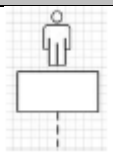
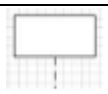
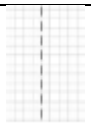
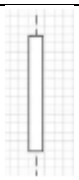
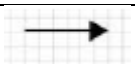
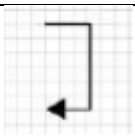
No.	Simbol	Nama	Keterangan
1.		*Action*	*State* dari sistem yang mencerminkan suatu aksi.
2.		*Decision*	Digunakan untuk menggambarkan suatu pengambilan keputusan / tindakan pada kondisi tertentu.
3.		*Initial Node*	Menunjukkan dimulainya suatu aktifitas.
4.		*Final Node*	Menunjukkan akhir dari aktifitas.
5.		*Fork Node*	Digunakan untuk menunjukkan aktifitas yang dilakukan secara paralel.
6.		*Join Node*	Digunakan untuk menunjukkan aktifitas yang digabungkan.
7.		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi.

2.15 *Sequence* Diagram

Sequence diagram digunakan untuk menunjukan aliran fungsionalitas dalam *use case*. *Sequence* diagram merupakan salah satu diagram *Interaction* yang

menjelaskan bagaimana suatu operasi itu dilakukan; message (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Tabel simbol sequenece diagram dapat dilihat pada Tabel 2.4 Simbol *Sequence* Diagram.

Tabel 2.4 Simbol *Sequence* Diagram

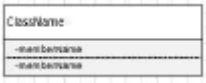
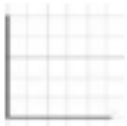
No.	Simbol	Nama	Keterangan
1.		Aktor	Menggambarkan peran pengguna ketika berinteraksi dengan sistem.
2.		*Object* (partisipan)	Merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal.
3.		*Lifeline*	Mengindikasikan keberadaan sebuah objek dalam basis waktu.
4.		*Activation*	Mengindikasikan sebuah objek yang akan melakukan sebuah aksi.
5.		*Message*	Mengindikasikan komunikasi antar *obejct*.
6.		*Self-Message*	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.
7.		*Alternative*	Mengambil keputusan/tindakan untuk suatu kondisi tertentu.

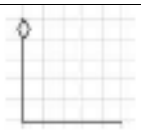
2.16 *Class* Diagram

Class diagram memberikan pandangan secara luas dari suatu sistem dengan

menunjukan kelas-kelasnya dan hubungan mereka. *Class* diagram bersifat statis, menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Class menggambarkan keadaan (*atribut*/properti). *Class* diagram menggambarkan struktur dan deskripsi *class*, *package*, atau objek berserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. Tabel simbol *class* diagram dapat dilihat pada Tabel 2.5 Simbol *Class* Diagram.

Tabel 2.5 Simbol *Class* Diagram

No	Simbol	Nama	Keterangan
1.		*Class*	Blok pembangunan pada pemrograman berorientasi objek. Bagian atas adalah bagian dari class. Bagian tengah mendefinisikan *property*/*atribut class*. Bagian akhir mendefinisikan *method* - *method* dari sebuah class.
2.		*Association*	*Relationship* paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapatmenampilkan hukum-hukum multiplisitas pada sebuah relasi.
3.		*Compisition*	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat bergantung tersebut.
4.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi

			elemen yang bergantung pada elemen yang tidak mandiri (*independent*).
5.		*Aggregation*	Mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi.
6.		*Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai *multiplicity*