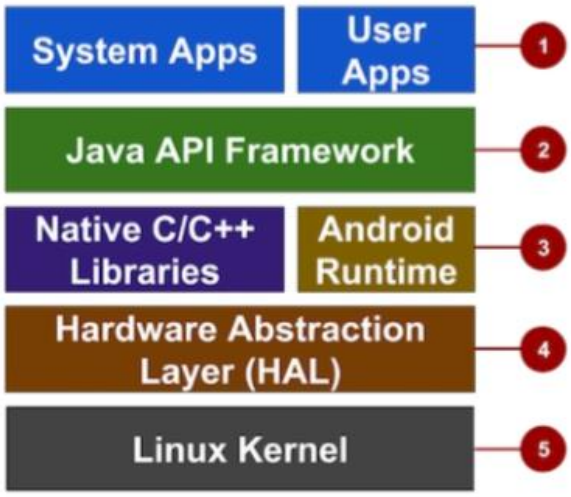
BAB II
LANDASAN TEORI

1.1 Tinjauan Pustaka

Pada pengembangan aplikasi *panic button* dibutuhkan beberapa komponen fisik pendukung baik berupa *hardware* maupun *software*. Selain komponen fisik, penyusun menggunakan komponen teoritis berupa pemodelan dengan *Unified Modeling Language (*UML), komponen tersebut berguna dalam mempermudah perancangan aplikasi agar sesuai dengan kaidah-kaidah yang telah ditetapkan.

1.1.1 Sistem Operasi Android

Android adalah system operasi dan platform pemrograman yang dikembangkan oleh Google untuk *smartphone* dan perangkat selular lainnya (seperti tablet). Android bias berjalan di bebrapa macam perangkat dari banyak produsen yang berbeda. Android menyertakan *development kit* perangkat lunak untuk penulisan kode asli dan perakitan modul perangkat lunak, untuk membuat aplikasi bagi pengguna android. Secara keseluuhan android menyertakan ekosistem untuk aplikasi selular. [3]



Gambar 2.1 Struktur Sistem Operasi Android

1. Aplikasi, aplikasi berada pada tingkat ini, bersama dengan aplikasi system inti untuk email, SMS, kalender, penjelajahan internet atau kontak.
2. Kerangka Kerja API Java, adalah semua fitur android yang tersedia untuk pengembang melalui antarmuka pemrograman aplikasi
3. Pustaka dan Waktu Proses Android: detiap aplikasi berjalan dalam prosesnya sendiri dan dengan *instance Android Runtime* sendiri, yang memungkinkan menjalankan beberapa mesin virtual sekaligus pada perangkat bermemori rendah. Android juga menyertakan rangkaian pustaka (*library) Android Runtime* yang menyediakan sebagian besar fungsionalitas Bahasa pemorgraman java, termasuk beberapa fitur bahasa Java 8 yang digunakan dalam kerangka kerja Java API.
4. Hardware Abstraction Layer (HAL), Lapisan ini menyediakan antarmuka standar yang menunjukan kemampuan perangkat keras di kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka masing-masing mengimplemtasikan antarmuka untuk komponen perangkat keras tertentu seperti modul kamera atau *Bluetooth.*
5. Kernel Linux, adalah fondasi platform Android. Lapisan di atasnya mengandalkan kernel linux untuk fungsionalitas pokok seperti *threading* dan manajemen memori tingkat rendah menggunakan kernel linux, memungkinkan Android memanfaatkan fitur keamanan utama dan memungkinkan produsen perangkat mengembangkan *driver* perangkat keras untuk kernel yang cukup dikenal.

1.1.2 People Nearby

People Nearby adalah salah satu fitur yang dikembangkan oleh *LINE Corporation.* Fitur ini memungkinkan pengguna *Line* untuk mencari teman baru dengan cara menampilkan daftar pengguna lain yang lokasinya berdekatan. Sementara fitur ini baru bisa digunakan di aplikasi Line yang berada di *smartphone* dan belum bisa digunakan pada *Line* untuk PC. [4]

Dimana fitur tersebut memiliki parameter utama yaitu ‘locationbias’. Parameter tersebut berfungsi sebagai hasil pencarian dari suatu objek pada area yang ditentukan, dengan menentukan baik radius plus lat / lng (*latitude/longitude)*, atau dua pasangan lat / lng mewakili titik-titik persegi panjang. Jika parameter ini tidak ditentukan, API menggunakan penyangkalan alamat IP secara default.

Ada beberapa cara ‘locaionbias’ pada fitur *People Nearby* agar dapat mengetahui posisi kita dengan parameter yang akan dijelaskan dibawah ini:

1. Bias IP: Menginstruksikan API untuk menggunakan pengalihan alamat IP. melewati string ‘ipbias’ (opsi ini tidak memiliki parameter tambahan).
2. Dengan menggunakan notasi ‘`Point:`’ Koordinat lat / lng tunggal. Gunakan format: `point:lat,lng`.
3. Dengan menggunakan variabel ‘`circular:`’ Sebuah string yang menentukan radius dalam meter, ditambah lat / lng dalam derajat desimal. Misalnya : `circle:radius@lat,lng`
4. Dengan menggunakan variable ‘`Rectangular:`’ Sebuah string yang menentukan dua pasangan lat / lng dalam derajat desimal, mewakili titik selatan / barat dan utara / timur dari sebuah persegi panjang. Gunakan format berikut: `rectangle:south,west|north,east`

1.1.3 Firebase Realtime Database (FRD)

Firebase Realtime Database adalah database *NoSQL* yang di-host pada cloud dan dapat digunakan untuk menyimpan dan menyinkronkan data antarpengguna secara *realtime.* Sinkronisasi *realtime* memudahkan untuk mengakses data dari perangkat apa pun: web atau telepon seluler, dan memudahkan untuk saling berkolaborasi.[5]

Firebase Realtime Database (FRD) memungkinkan pengembang 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. Oleh karena itu, perlu dipikirkan bagaimana pengguna mengakses data, kemudian buat struktur data sesuai dengan kebutuhan tersebut.

Gambar 2.2 Struktur Firebase Realtime Database

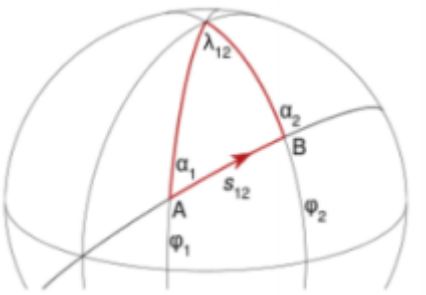
Berdasarkan dari gambar di atas FRD menggunakan format Json (*javascript format notation)* dimana data disimpan pada simpul-simpul yang berisi object dari turunannya. FRD dikatakan *realtime* dikarenakan data yang baru masuk akan menimpa data yang lama sehingga proses pengolahan data menjadi lebih ringan

oleh karena itu data lama yang tergantikan akan terhapus secara permanent sehingga tidak membebani kinerja system.

Panic Button memanfaatkan fitur *realtime* yang dimiliki FRD sebagai sarana penyimpanan data lokasi pengguna secara *realtime*. Data lokasi tersebut secara kontinyu akan ter-update secara otomatis ketika pengguna mengalami perubahan pada titik koordinat, dengan kata lain data akan ter-update ketika pengguna berpindah tempat.

1.1.4 Vincenty Formulae

Metode Vincenty untuk penentuan jarak antara dua titik dikembangan oleh [6], mengasumsikan bentuk bumi yang mendekati ellipsoid. Penggunaan ellipsoid tersebut menyebabkan hasil perhitungan jarak akan lebih mendekati yang sebenarnya daripada metode lain yang mengasumsikan bentuk bumi datar maupun bentuk bumi bola. Ilustrasi jarak ditunjukkan pada Gambar 2.6.



Gambar 2.3 Ilustrasi Azimuth Metode Vincenty

Penentuan azimuth pada metode Vincenty menggunakan prinsip *inverse problem*, dimana diperlukan dua set koordinat yang dinotasikan dalam sistem koordinat geodetik (λ,ϕ). Penentuan azimuth dengan menggunakan notasi *Vincenty* dapat didefinisikan sebagai berikut:

1.1.5 Google Maps Android API V2

Google menyediakan *API* (*Aplication Programming Interface*) yang stabil dan komprehensif untuk pengembangan aplikasi yang berbasis Android khususnya bagi pengembang yang menggunakan *Google Maps Services* dalam aplikasi mereka. Menggunakan API ini pengembang dapat dengan mudah mengintegrasikan peta ke aplikasi mereka dan *API* secara otomatis menangani akses ke server *Google Maps*, pengunduhan data, tampilan peta, dan respon terhadap gerakan. Selain itu *API* ini juga dapat digunakan untuk menambahkan beberapa atribut kedalam peta misalnya:

1. *Markers,* atribut yang digunakan untuk menunjukan posisi spesifik didalam peta.
2. *Line Segment* (*Polylines*), adalah atribut yang digunakan untuk menunjukan route dari lokasi spesifik.
3. *Enclosed Lines (Polygon),* adalah atribut yang digunakan untuk menandai wilayah secara spesifik berdasarkan radius yang telah ditentukan. dan
4. Overlays, adalah berbagai macam gambar yang dapat ditampilkan didalam peta seperti *zoom controls, compass* dan lain-lain.

Untuk menggunakan layanan *Google Maps API* pengembang harus memilki *API KEY* yang telah terdaftar di laman proyek google. *API KEY* didapatkan dengan cara mendaftarkan *Signing Report* (terenkripsi SHA1 atau MD5) pada aplikasi yang sedang dibangun untuk mendapatkan kunci unik yang digunakan sebagai *project id.*[7]

1.1.6 Panic Button

Aplikasi *panic button* atau bisa juga disebut tombol darurat sangat berguna untuk menghadapi berbagai situasi yang membutuhkan penanganan cepat. Mengintegrasikan *panic button* ke sistem keamanan akan meningkatkan *awareness* situasional petugas keamanan dan membantunya menangani situasi mendesak lebih cepat.[8]

Hal ini sejalan dengan konsep yang di ajukan oleh Kintronics dalam postingannya pada blognya. "Keadaan darurat terjadi tanpa adanya peringatan. Ini bisa menjadi masalah keselamatan, seperti situasi penembakan atau serangan kriminal. Panic button ini bisa memberikan pemberitahuan darurat yang cepat."[9]

Secara umum aplikasi *panic button* memiliki 2 kemampuan utama yaitu:

1. Aplikasi mampu mengirim pesan darurat umumnya pesan tersebut berisi data lokasi pengguna secara *realtime*
2. Aplikasi memiliki tempat penyimpanan kontak pengguna yang akan menerima pesan darurat.

Aplikasi *Panic Button* memiliki banyak turunannya dengan banyak fitur yang berbeda pada setiap turunannya, tetapi terdapat satu fitur yang wajib ada pada aplikasi *Panic Button* yaitu fitur *emergency broadcast message*. Fitur tersebut wajib ada pada setiap turunan aplikasi *Panic Button* dikarenakan fitur tersebut adalah fungsi utama dalam aplikasi.

1.1.7 GeoFire

GeoFire adalah pustaka dukungan yang *open source* untuk platform Android yang memungkinkan untuk menyimpan dan meminta set kunci berdasarkan lokasi geografisnya. Pada intinya, GeoFire hanya menyimpan lokasi dengan kunci string. Namun manfaat utamanya adalah memungkinkan untuk mengambil kueri kunci pada geografis tertentu secara *realtime*.

GeoFire memiliki beberapa kelas yang digunakan dalam pengembangan aplikasi *Panic Button* diantaranya:

1.1.7.1 GeoLocation

GeoLocation adalah salah satu kelas dalam pustaka dukungan *GeoFire* yang berfungsi sebagai pengolah data geografis yang disimpan di dalam FRD. Data tersebut diolah sebegai data pendukung dalam menentukan posisi realtime yang digunakan sebagai *object reference* pada metode *setLocation()*.[10]

Kelas ini akan digunakan pada aplikasi yang akan dikembangkan sebagai pengambil data koordinat dari GPS, data tersebut akan diolah menjadi *marker* pada laman peta. *Marker* ini berfungsi sebagai penanda lokasi pada laman peta, terdapat empa jenis penanda lokasi yang memiliki logo dan maksud yang berbeda hal ini akan dijelaskan secara rinci pada bab iii.

1.1.7.2 GeoQuery

GeoQuery adalah salah satu kelas dalam pustaka dukungan *GeoFire* yang berfungsi sebagai filter pencarian terhadap suatu object yang memiliki data geografis berupa koordinat gps. Objek tersebut dicari berdasarkan radius yang telah diatur sebelumnya.

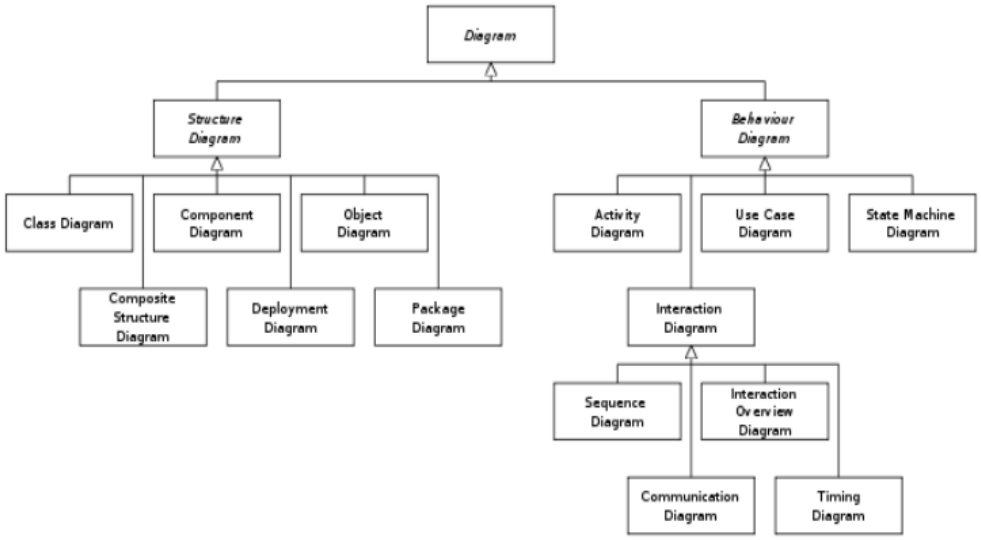
Kelas ini digunakan dalam pengembangan aplikasi *Panic Button* sebagai pencari pengguna yang menekan tombol darurat sehingga pengguna tersebut dapat diketahui oleh pengguna lainnya. Penekan tombol darurat tersebut akan diketahui oleh pengguna sekitarnya karena filter yang telah diatur pada system berdasarkan radius terdekat. Radius tersebut dapat berubah-ubah tergantung ada tidaknya pengguna lain yang berada disekitar penekan tombol darurat tersebut.

1.2 Perancangan Sistem Dengan UML

UML adalah kumpulan notasi grafis yang membantu dalam mengembangkan dan merancang sistem perangkat lunak, khususnya sistem perangkat lunak yang dibangun dengan object oriented. Pemodelan sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Berdasarkan pendapat yang dikemukakan di atas dapat ditarik kesimpulan bahwa *Unified Modelling Language* (UML) adalah sebuah bahasa pemodelan yang merepresentasikan dan

memvisualkan sebuah sistem pengembangan perangkat lunak berbasis object oriented.[12]

UML memberikan 13 jenis diagram yang digunakan untuk merepresantasikan bagian penting dari sebuah rancangan perangkat lunak. Penggunaan jenis diagram didasarkan pada kebutuhan dan kerakteristik sebuah perangkat lunak. Berikut ini adalah jenis-jenis diagram beserta fungsinya:



Gambar 2.4 Klasifikasi Diagram UML

Pada gambar di atas terdapat berbagai macam Diagram *UML* namun pada kesmpatan kali ini penulis menggunakan hanya bebrapa untuk mempermudah dalam perancangan sistem secara visual.

Tabel 2.1 Deskripsi Diagram UML

Diagram	**Kegunaan**	**Learning Priority**
Activity	Menggambarkan proses aplikasi tingkat tinggi, termasuk aliran data, atau untuk model logika yang kompleks dalam sistem	*High*

Class	Menunjukkan sekumpulan elemen model statis seperti kelas dan jenis, isinya, dan hubungan mereka.	*High*
Communication	Menunjukkan contoh dari kelas, hubungan antar kelas, dan alur pesan antara mereka	*Medium*
Component	Menggambarkan komponen yang membentuk sebuah aplikasi	*Medium*
Composite structure	Menggambarkan struktur internal dari sebuah pengklasifikasi	*Low*
Deployment	pemindahan artifak ke node	*Medium*
Interaction overview	Sebuah varian dari diagram aktivitas yang mengontrol aliran dalam proses sistem	*Low*
Object	Menggambarkan objek dan relasi mereka di sebuah titik waktu tertentu	*Low*
Package	Menunjukkan bagaimana elemen model akan disusun dalam paket maupun dependensi antara bentuk paket	*Low*
Sequence	Interaksi antar obyek, penekanan pada sequence	*High*
State machine	Menjelaskan lingkungan sebuah objek atau interaksi di dalamnya, maupun transisi antara lingkungan	*Medium*
Timing	Menggambarkan perubahan keadaan dari waktu ke waktu	*Low*

Use case	Bagaimana pengguna berinteraksi dengan sistem	*Medium*

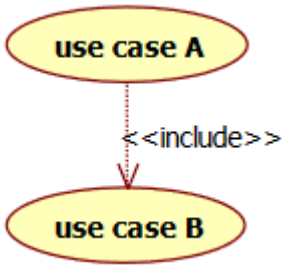
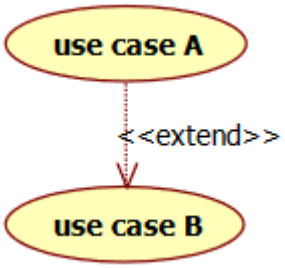
Dalam pengembangan aplikasi permainan Edunvi, penelitihanya akan menggunakan 4 macam diagram UML yaitu *Use Case Diagram, Class Diagram, Sequence Diagram* dan *Activity Diagram*. Pemilihan diagram ini didasarkan pada tingkat *Learning Priority* yang tinggi serta dirasa cukup mewakili dari seluruh segmentasi pada diagram UML. Notasi diagram-diagram tersebut adalah sebagai berikut:

1.2.1 Use Case Diagram

Use case adalah sebuah kegiatan yang dilakukan oleh sistem yang biasanya menanggapi permintaan dari pengguna sistem. Dengan kata lain usecase diagram secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna mengharapkan interaksi dengan sistem itu.[13] Setiap *use case* nantinya akan didefinisikan lebih lanjut melalui sebuah skenario. Notasi *Use case* adalah sebagai berikut:

Tabel 2.2 Notasi Use Case

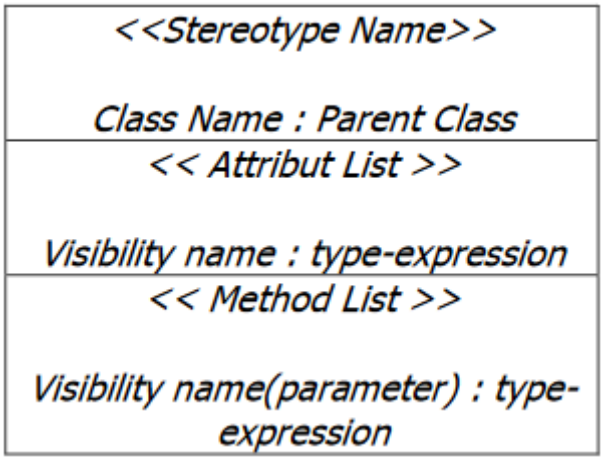
Simbol	Nama	Kegunaan
	*Actor*	Aktor adalah segala hal diluar sistem (bisa manusia, sistem, atau perangkat) yang akan menggunakan sistem tersebut untuk melakukan sesuatu.
	Dependency	*Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah

		adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah. Terdapat dua tipe yaitu *include* dan *extend.*
	*Include*	Menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Use case* A dapat berjalan jika *use case* B sudah dijalankan minimal satu kali.
	*Extends*	Menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada di garis dengan panah. *Use case* A memanggil *use case* B pada kondisi tertentu.
	*Generalization*	Menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan generalization, classyang lebih spesifik (subclass) akan menurunkan atribut dan operasi dari classyang lebih umum (superclass)
	Association	Mengidentifikasikan interaksi antara setiap aktor tertentu dengan setiap use case tertentu. Digambarkan sebagai garis antara aktor terhadap use caseyang bersangkutan.
	System	Menyatakan batasan sistem dalam relasi dengan aktor-aktor yang menggunakannya (di luar sistem) dan fitur-fitur yang harus

		disediakan (dalam sistem). Sistem disertai label yang menyebutkan nama dari sistem.
	*Use case*	Mengidentifikasi fitur kunci dari sistem. Tanpa fitur ini, sistem tidak akan memenuhi permintaan *user/actor*. Setiap use casemengekspresikan tujuan dari sistem yang harus dicapai dan diberi nama sesuai dengan tujuannya.
	*Interaction*	*Interaction* digunakan untuk menunjukkan baik aliran pesan atauinformasi antarobyek maupun hubungan antar obyek. Biasanya *interaction* ini dilengkapi juga dengan teks bernama operation signature yangtersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan

1.2.2 Class Diagram

Class Diagram adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun[14]. Notasi *Class Diagram* adalah sebagai berikut:



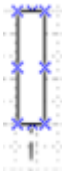
Gambar 2.5 Notasi Class Diagram

1.2.3 Sequence Diagram

Sequence diagram adalah diagram yang digunakan untuk mendefinisikan input dan output serta urutan interaksi antara pengguna dan sistem untuk sebuah *use case*.[15] Notasi sequence diagram adalah sebagai berikut:

Gambar 2.6 Notasi Sequence Diagram

Simbol	Nama	Kegunaan
: objek	*Object*	Merupakan *instance* dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.
	*Actor*	Aktor adalah segala hal diluar sistem (bisa manusia, sistem, atau perangkat) yang akan menggunakan sistem tersebut untuk melakukan sesuatu

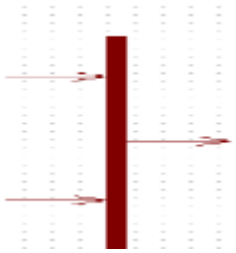
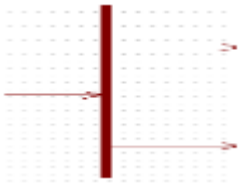
	Lifeline	*Lifeline* mengindikasikan keberadaan sebuah objectdalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.
	*Activation*	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. Activation mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.
	*Message*	*Message* digambarkan dengan anak panah horizontal antara *Activation. Message* mengindikasikan komunikasi antara object-object.

1.2.4 Activity Diagram

Activity Diagram adalah sebuah alur kerja yang menjelaskan berbagai kegiatan pengguna (atau sistem), orang yang melakukan aktivitas, dan aliran sekuensial dari aktivitas-aktivitas tersebut. *Activity diagram* dapat juga digunakan untuk objek memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari aksi tersebut.[16] Notasi pada *Activity Diagram* adalah:

Tabel 2.3 Notasi Activity Diagram

Simbol	Nama	Kegunaan
	*Start Point*	Menunjukan dimana aliran kerja dimulai.

	*End Point*	Menunjukan dimana aliran kerja berakhir
	*Activity*	Menunjukan kegiatan dalam aliran kerja
	*Join*	Menunjukan percabangan pada aliran kerja
	*Fork*	Menunjukan penggabungan dalam aliran kerja

1.3 Entity Relationship Diagram

Basis data atau kerap disebut *database* merupakan kumpulan informasi yang disimpan secara sistematis dalam perangkat komputer sehingga dapat dicari dan diperiksa melalui suatu program komputer saat informasi tertentu sedang dibutuhkan.[17]

Oleh karena itu, agar menjadi sistem *database* yang rapi dan terstruktur, penulis menggunakan *Entity Relationship Diagram* (ERD), yaitu sebuah model untuk menyusun *database* agar dapat menggambarkan data yang mempunyai relasi dengan *database* yang akan didesain.

Diagram ER biasanya berhubungan langsung dengan diagram *data flow* untuk menampilkan konten *data store*. Ketiga hal tersebut dapat membantu memvisualisasikan bagaimana data saling terhubung dan berguna untuk mengonstruksi basis data relasional. Terdapat beberapa istilah umum dan simbol ERD serta komponen penyusun ERD seperti yang akan dijelaskan dibawah ini:

1.3.1 Entitas

Kumpulan objek yang dapat diidentifikasikan secara unik atau saling berbeda. Simbol dari entitas biasanya digambarkan dengan persegi panjang. Selain itu, ada juga “Entitas Lemah” yang dilambangkan dengan gambar persegi panjang kecil di dalam persegi panjang yang lebih besar. Disebut entitas lemah karena harus berhubungan langsung dengan entitas lain sebab dia tidak dapat teridentifikasi secara unik.

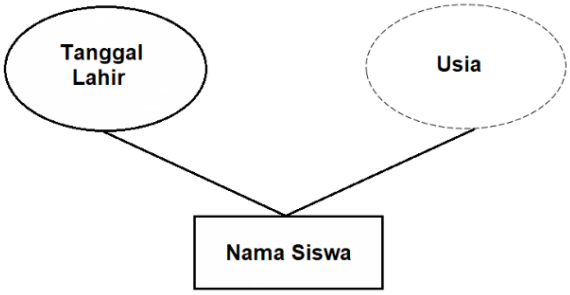
Gambar 2.7 Simbol Entitas

1.3.2 Atribut

Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Atribut kunci merupakan hal pembeda atribut dengan entitas. Gambar atribut diwakili oleh simbol elips dan terbagi menjadi beberapa jenis:

1. Atribut kunci (*key*): atribut yang digunakan untuk menentukan entitas secara unik. Contoh: NPWP, NIM (Nomor Induk Mahasiswa).
2. Atribut simpel: atribut bernilai tunggal yang tidak dapat dipecah lagi (*atomic*). Contoh: Alamat, tahun terbit buku, nama penerbit.
3. Atribut multinilai (*multivalue*): atribut yang memiliki sekelompok nilai untuk setiap entitas instan. Contoh: nama beberapa pengarang dari sebuah buku pelajaran.
4. Atribut gabungan (composite): atribut yang terdiri dari beberapa atribut yang lebih kecil dengan arti tertentu. Contoh: nama lengkap yang terbagi menjadi nama depan, tengah, dan belakang.

5. Atribut derivatif: atribut yang dihasilkan dari atribut lain dan tidak wajib ditulis dalam diagram ER. Contoh: usia, kelas, selisih harga.

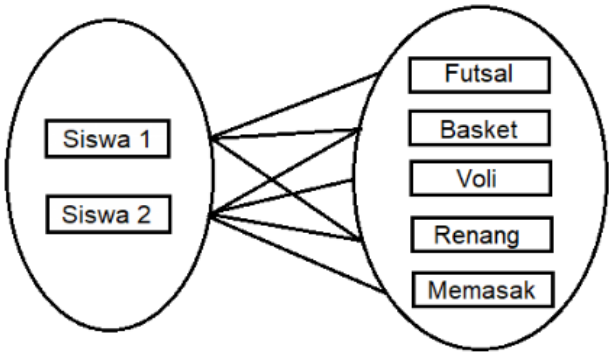


Gambar 2.8 Simbol Atribut Derivatif

1.3.3 Relasi

Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Gambar relasi diwakili oleh simbol belah ketupat. Relasi juga terbagi menjadi beberapa jenis:

1. *One to one*, setiap entitas hanya bisa mempunyai relasi dengan satu entitas lain. Contoh: siswa dengan nomor induk siswa
2. *One to many*, hubungan antara satu entitas dengan beberapa entitas dan sebaliknya. Contoh: guru dengan murid dan sebaliknya.
3. *Many to many*, setiap entitas bisa mempunyai relasi dengan entitas lain, dan sebaliknya. Contoh: siswa dan ekstrakurikuler.



Gambar 2.9 Contoh Relasi