BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori adalah konsep yang disusun secara sistematis serta memiliki dasar yang kuat dalam sebuah penelitian. Tujuan dari landasan teori yaitu memberikan gambaran dari teori-teori yang terkait dalam penelitian. Landasan teori yang digunakan untuk membangun aplikasi *Smart Motorcycle Assistant* ini meliputi penjelasan tentang Sensor *Accelerometer*, *Global Positioning System* (GPS), Aplikasi *Mobile*, *Android*, Kotlin, JavaScript, NodeJs, *Extensible Markup Language* (XML), *Application Programming Interface* (API), JavaScript *Object Notation* (JSON), Google Maps API, Google Directions API, Google Places API, Amazon Web Service, *Web Service*, XAMPP, Basis Data MySQL (*Database*), *Advenced Encryption Standard* (AES), *Unified Modeling Language* (UML), dan lain-lain.

2.2 Sepeda Motor

Sepeda motor merupakan sebuah kendaraan roda dua yang biasa digunakan sebagai kendaraan pribadi atau kendaraan umum di berbagai negara di seluruh dunia. Sepeda motor memiliki komponen-komponen yang dapat membuatnya bergerak, diantaranya seperti mesin, sistem pengereman, roda, dan lain-lain. Tipe, jenis, merek, ukuran pada sepeda motor memiliki banyak variasi. Beberapa jenis sepeda motor diantaranya adalah sepeda motor bebek (*underbone*), sepeda motor *automatic*, dan sepeda motor *sport*. Berdasarkan data yang didapatkan dari Badan Pusat Statistik Indonesia, data jumlah kendaraan sepeda motor di Indonesia pada tahun 2021 adalah 121.209.304 [1].

Tentunya untuk menjalankan sepeda motor diperlukan sumber tenaga seperti bahan bakar bensin, listrik, dan sebagainya. Pada penelitian dan pembangunan sistem yang akan dilakukan, jenis sepeda motor yang akan diteliti yaitu sepeda motor yang menggunakan bahan bakar bensin. Berdasarkan data spesifikasi sepeda motor yang didapatkan dari beberapa *website* resmi perusahaan besar yaitu Honda, Yamaha, dan Kawasaki dan hasil pengujian konsumsi bahan bakar dari *website* autofun yang beralamat “https://www.autofun.co.id” yang

diakses pada tanggal 9 Mei 2023. Berikut daftar spesifikasi dan konsumsi bahan bakar pada sepeda motor dapat dilihat pada Tabel 2.1 berikut:

Tabel 2.1 Daftar Spesifikasi Sepeda Motor

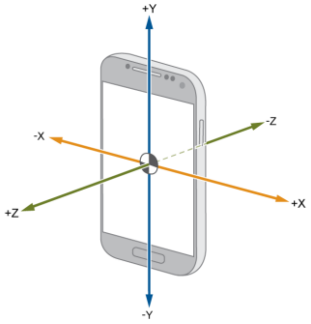
No.	Merek	Tipe	Jenis	Kapasitas Tangki (L)	Konsumsi Bensin (Km/L)
1	Honda	Beat	Automatic	4.2	60.6
2	Honda	Beat Street	Automatic	4.2	60.6
3	Honda	Genio	Automatic	4.2	59.1
4	Honda	Scoopy	Automatic	4.2	59
5	Honda	Vario 125	Automatic	5.5	50.2
6	Honda	Vario 150	Automatic	5.5	46.9
7	Honda	Vario 160	Automatic	5.5	48
8	Honda	PCX 160	Automatic	8.1	42
9	Honda	ADV	Automatic	8.1	42.15
10	Honda	Sonic 150R	Sport	4	45.35
11	Honda	CB150R Streetfire	Sport	12	46
12	Honda	CRF150L	Sport	7.2	42.3
13	Honda	CBR150R	Sport	12	43
14	Honda	CBR250RR	Sport	14.5	25
15	Honda	CRF250 Rally	Sport	12.8	33.4
16	Honda	CRF250L	Sport	7.8	35.8
17	Honda	Verza	Sport	12.2	43.5
18	Honda	Revo	Underbone	4	59.8
19	Honda	Supra X 125 Fi	Underbone	4	57
20	Honda	Supra GTR 150	Underbone	4.5	45
21	Yamaha	XMAX	Automatic	13	31.4
22	Yamaha	NMAX	Automatic	7.1	39.4
23	Yamaha	Aerox	Automatic	5.5	38
24	Yamaha	Lexi	Automatic	4.2	42.3
25	Yamaha	X-Ride	Automatic	4.2	49
26	Yamaha	Gear 125	Automatic	4.2	44.5
27	Yamaha	Mio M3	Automatic	4.2	50
28	Yamaha	Fino 125	Automatic	4.2	47
29	Yamaha	R15	Sport	11	42.6
30	Yamaha	R25	Sport	14	23.2
31	Yamaha	V-Ixion	Sport	12	40.7
32	Yamaha	V-Ixion R	Sport	11	52
33	Yamaha	Mx King	Underbone	4.2	50
34	Yamaha	Jupiter Z1	Underbone	4.1	52.9
35	Yamaha	Vega	Underbone	4	55
36	Kawasaki	Ninja 250	Sport	14	25
37	Kawasaki	KLX 250	Sport	7.7	30
38	Kawasaki	KLX 150	Sport	6.9	31.8

2.3 Keselamatan Berkendara

Keselamatan dalam berkendara merupakan aspek utama dalam berkendara. Pemahaman tentang aturan-aturan yang harus dipatuhi agar mencegah kecelakaan itu terjadi seperti memahami arti dari tanda-tanda jalan serta praktik-praktik lainnya. Berdasarkan pengujian *flicker* yang dilakukan, didapatkan hasil kemungkinan kesempatan beristirahat yang paling sesuai dengan kebanyakan data yang diperoleh adalah tiga jam atau jarak tempuh mencapai 120 Km jika mengemudi dengan kecepatan rata-rata 40km/jam. Hasil tersebut merupakan hasil rata-rata dari pengemudi dengan usia 40 tahun ke bawah [11]. Salah satu aturan yang ditetapkan yaitu pengguna sepeda motor disarankan untuk beristirahat saat merasa kelelahan untuk mengurangi risiko kecelakaan. Berdasarkan Undang-Undang No. 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan (LLAJ), pada pasal 90 ayat (3) bahwa pengguna sepeda motor diwajibkan untuk beristirahat paling sedikit 30 menit setelah berkendara selama 4 (empat) jam. Namun, interval waktu untuk beristirahat dapat disesuaikan dengan kondisi fisik masing-masing tergantung kesehatan fisik, usia, dan faktor-faktor lainnya.

2.4 *Accelerometer*

Sensor *Accelerometer* merupakan sensor yang digunakan untuk mengukur kecepatan dan percepatan suatu objek. Pengukuran percepatan menggunakan sensor ini dapat mengukur dan mengidentifikasi secara statis maupun dinamis. Sensor ini bekerja berdasarkan hukum fisika di mana pengukuran secara statis dilakukan dengan mengukur terhadap gravitasi bumi. Sedangkan pengukuran secara dinamis yaitu mengukur terhadap percepatan atau perpindahan posisi suatu objek. Pengukuran sensor *Accelerometer* pada *smartphone* menggunakan tiga sumbu utama yaitu sumbu x, sumbu y, dan sumbu z. Sensor akan memprediksi ketiga sumbu tersebut dan akan mendeteksi perpindahan atau percepatan jika melebihi *threshold* atau ambang batas yang ditentukan [12]. Nilai dari setiap sumbu dihitung menggunakan rumus *Pythagoras* untuk mendapatkan nilai magnitudo dengan satuan m/s^2. $Magnitudo = \sqrt{x^2 + y^2 + z^2}$ merupakan rumus untuk mencari magnitudo dari total percepatan yang dideteksi oleh sensor. Berikut merupakan gambaran dari ketiga sumbu *Accelerometer* yang terdapat pada *smartphone* dapat dilihat pada Gambar 2.1.



Sumber Gambar: https://pelajar.net/sensor-*Accelerometer*/

Gambar 2.1 *Sensor Accelerometer*

Melalui ketiga sumbu tersebut, perangkat yang memiliki sensor *Accelerometer* dapat mengidentifikasi gerakan seperti guncangan, getaran, dan perputaran yang dapat diimplementasikan dengan tujuan tertentu.

2.5 *Global Positioning System* (GPS)

Global Positioning System (GPS) adalah sebuah sistem satelit yang menyediakan informasi geolokasi dan waktu. GPS merupakan sebuah sistem navigasi atau penentu posisi yang memberikan posisi dan kecepatan tiga dimensi secara kontinu. Sinyal pengukuran GPS terdiri dari waktu saat sinyal tersebut dikirimkan dan waktu saat sinyal tersebut diterima. GPS tidak mengharuskan penggunanya untuk mengirim data apa pun, melainkan bekerja secara independen dari sinyal yang berasal dari *smartphone* seperti internet ataupun telepon. Dengan adanya GPS maka pengguna dapat mengetahui lokasi keberadaannya hingga dapat melacak keberadaan pengguna lain [13].

GPS dapat mengukur jarak antara penggunanya dengan satelit lalu dihitung dengan selisih waktu yang didapatkan antara pengirim dan penerima yaitu satelit dan perangkat penerima sinyal. Dengan membandingkan waktu diterimanya sinyal tersebut dari beberapa satelit maka GPS dapat menentukan posisi penerima [13]. GPS akan memberikan data berupa koordinat "*latitude*, *longitude*" yang dapat digunakan untuk mengetahui posisi dari perangkat.

2.6 Aplikasi *Mobile*

Aplikasi *mobile* adalah aplikasi dari sebuah perangkat lunak yang digunakan pada perangkat mobile seperti *Smartphone* atau Tablet. Terdapat dua jenis pada aplikasi *mobile* yaitu *mobile native* dan *mobile* web. Aplikasi *mobile*

native merupakan aplikasi yang dibangun khusus untuk *mobile* dengan sistem operasi seperti iOS atau *Android*. Sedangkan untuk aplikasi *mobile* web aplikasi yang berinteraksi melalui browser namun dirancang untuk *mobile* dan membutuhkan koneksi internet.

Aplikasi *mobile* dapat berinteraksi dengan berbagai fitur yang tersedia pada *smartphone* seperti GPS, kamera, sensor akselerometer, dan fitur-fitur lainnya. Dengan banyaknya fitur yang tersedia tentunya aplikasi *mobile* dapat membantu dan menjadi bagian penting dari kehidupan sehari-hari sehingga aplikasi *mobile* menjadi teknologi yang paling banyak digunakan [14].

2.7 *Android*

Android merupakan sebuah sistem operasi untuk perangkat *mobile* pertama pada tahun 2008 yang diperkenalkan oleh Google. *Android* bersifat *open-source* yang di mana pengguna dapat mengakses dan memodifikasi kode dengan bebas. Banyak *developer* dalam suatu organisasi yang melakukan modifikasi sistem operasi *Android* seperti LineageOS yang merupakan sistem operasi kustom berbasis Linux dengan menambahkan banyak fitur tambahan. *Android* dirancang untuk beberapa perangkat seperti *smartphone*, *smartwatch*, tablet, *smart* TV, dan sebagainya.

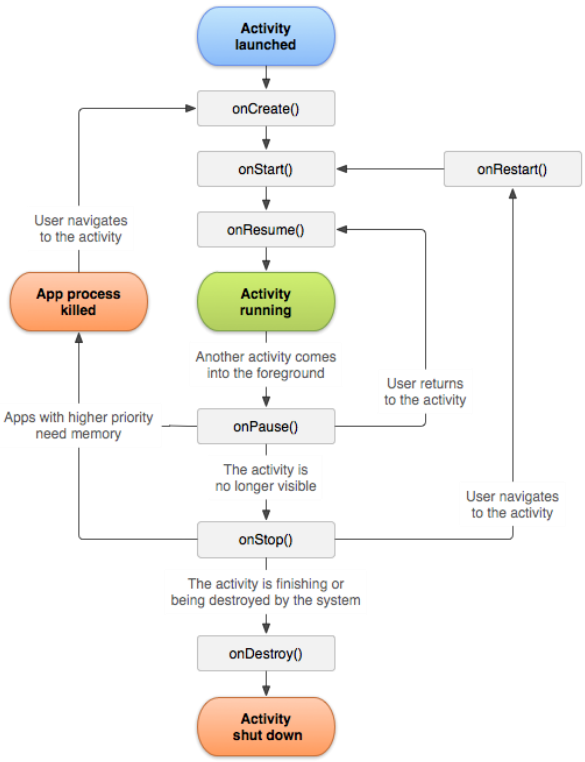
Sistem operasi *Android* terus berkembang dari tahun ke tahun, dari muali versi pertama yang dikenalkan yaitu *Android* 1.0 (*Alpha*) hingga *Android* terbaru saat ini yaitu *Android* 13 (*Tiramisu*) yang mempunyai segudang fitur seperti *Material Design* atau *Material* U yang pertama kali diperkenalkan pada *Android* 12 (*Snow Cone*) dan berfokus pada sistem keamanan untuk penggunanya.

Sistem *Android* menggunakan kernel Linux yang mempunyai arsitektur berbasis aplikasi yang memungkinkan untuk menjalankan lebih dari satu aplikasi secara bersamaan. *Android* juga memiliki SDK atau *Software Development Kit* yang memungkinkan pengembang untuk membuat dan mengembangkan aplikasi *Android* menggunakan bahasa pemrograman seperti Java atau Kotlin.

Aplikasi yang dijalankan pada *Android* berformat *Android Package Kit* (APK) yang di dalamnya berisi semua kode program, data dan sumber daya yang diperlukan. Penggunanya dapat memasang aplikasi dengan berbagai cara seperti melalui toko resmi Google Play Store atau sumber lain [15].

2.7.1 *Android Life Cycle*

Android Life Cycle merupakan siklus hidup pada aplikasi yang digunakan oleh sistem operasi *Android* untuk mengatur dan mengelola aktivitas pada aplikasi. Aktivitas merupakan komponen utama yang menunjukkan *interface* dari aplikasi. Aktivitas memiliki siklus hidup yang terdiri dari serangkaian metode yang digunakan oleh sistem sesuai dengan status dari aktivitas tersebut. Pengembang aplikasi berbasis *Android* perlu memahami konsep dari siklus hidup ini untuk membangun aplikasi yang mempunyai performa baik, minim kesalahan, dan stabil. Selain itu, pengembang juga harus mengetahui jika terjadi perubahan pada siklus hidup seperti rotasi layar, masuk atau keluar dari suatu aktivitas, dan sebagainya. Berikut ilustrasi sederhana dari siklus proses aktivitas [15] dapat dilihat pada Gambar 2.2.



Sumber Gambar:
https://developer.*Android*.com/guide/components/activities/activity-lifecycle?hl=id#alc

Gambar 2.2 Ilustrasi *Android Life Cycle*

Penjelasan pada gambar mengenai *Android Life Cycle* dari mulai aktivitas dibuat hingga aktivitas dimatikan adalah sebagai berikut:

1. ***onCreate***

Metode yang akan dijalankan saat awal aktivitas dijalankan atau sebuah aktivitas dibuat.

2. *onStart*

Metode yang dijalankan setelah metode *onCreate()* selesai dieksekusi. Pada metode ini aktivitas dapat dilihat oleh pengguna aplikasi.

3. *onPause*

Metode yang terjadi karena pengguna aplikasi meninggalkan aktivitas atau menghentikan aplikasi secara sementara karena pengguna membuka aplikasi lain. Metode ini tidak menjalankan perintah apa pun dan aktivitas masih bisa berjalan di belakang layar.

4. *onResume*

Metode untuk memasuki atau melanjutkan aktivitas yang sebelumnya dihentikan sementara. Metode aplikasi ini juga bisa disebut mengembalikan aktivitas yang berada pada belakang layar ke latar depan agar dapat terlihat oleh pengguna.

5. *onStop*

Metode yang dijalankan baik secara paksa atau tidak oleh sistem *Android* saat aktivitas tidak dapat dilihat oleh pengguna. Metode ini digunakan untuk menghentikan aktivitas yang sudah tidak perlu dijalankan saat komponen-komponen tidak terlihat pada layar.

6. *onDestroy*

Metode yang dipanggil sebelum aktivitas ditutup dan akan menghapus aktivitas dan komponen-komponen di dalamnya. Metode ini terjadi saat pengguna keluar aplikasi dan menghapus aplikasi dari belakang layar.

2.7.2 *Android* Studio

Android Studio adalah *software Integrated Development Environment* (IDE) resmi untuk sistem operasi Andoid untuk pengembangan aplikasi *Android*. *Android* Studio dibangun oleh JetBrains yang dirancang khusus untuk pengembangan *Android*. *Android* Studio juga memungkinkan pengembang untuk membuat dan mengelola proyek *Android* yang kompleks, juga terdapat berbagai fitur seperti penyelesaian kode secara otomatis, *testing*, dan *debugging* [15].

Berdasarkan penelitian yang dilakukan bahwa para pengembang *Android* pada umumnya memiliki pengetahuan dan kesadaran yang terbatas dalam membangun aplikasi yang sesuai dengan model *lifecycle*. Hal tersebut tentunya memengaruhi performa aplikasi yang dibangun. *Android* Studio dapat mengelola kesesuaian *lifecycle* secara otomatis untuk menjaga performa aplikasi yang dibangun terutama bagi pengembang aplikasi *Android* baru yang masih kurang pengalaman [15].

2.7.3 *Android Architecture*

Arsitektur *Android* merupakan sebuah kerangka kerja yang digunakan untuk mengembangkan aplikasi *Android* dengan terstruktur. Dengan menggunakan arsitektur *Android*, pengembang lebih mudah dalam mengembangkan aplikasi *Android* karena dapat memisahkan tugas-tugas aplikasi menjadi beberapa lapisan serta mengurangi *duplicate code* [16]. Terdapat beberapa arsitektur yang sering digunakan dalam pengembangan aplikasi *Android* antara lain sebagai berikut:

1) Model-View-Controller (MVC), memisahkan kode menjadi tiga komponen utama yaitu model (data), *view* atau tampilan, dan *controller*.
2) Model-View-Presenter (MVP), memisahkan kode menjadi tiga komponen utama yaitu model (data), *view* atau tampilan, dan presenter yang bertugas sebagai pengendali antara data dan tampilan.
3) Model-View-ViewModel (MVVM), memisahkan kode menjadi tiga komponen utama yaitu model (data), *view* atau tampilan, dan ViewModel yang bertugas sebagai perantara antara tampilan dan model yang memungkinkan pengikatan data binding agar lebih mudah dan terstruktur.
4) *Clean Architecture*, membagi kode ke dalam beberapa lapisan yang terpisah seperti presentasi, *domain*, dan data. Setiap lapisan berkomunikasi yang dilakukan melalui *interface*.

Penggunaan arsitektur MVVM menjadikan pengembangan aplikasi lebih mudah karena terpisah dari *Android Lifecycle* yang memungkinkan untuk digunakan diberbagai tampilan yang memiliki struktur data yang sama. Secara keseluruhan, MVVM dapat mengurangi keterikatan kode, mengurangi kode duplikat [16].

2.7.4 *Services*

Services merupakan sebuah komponen aplikasi yang dapat beroperasi lama di *background* dan tidak menampilkan tampilan antarmuka. Komponen aplikasi ini akan terus berjalan di *background* meskupun pengguna mengeluarkan dan menutup aplikasi atau berpindah ke aplikasi lain. *Service* ini memungkinkan untuk menjalankan transaksi jaringan, media *player*, melkukan *input* atau *output*, dan lain sebagainya. Beberapa jenis *service* diantaranya adalah sebagai berikut:

1) *Foreground Service*

   Service yang memungkinkan untuk melakukan beberapa operasi yang terlihat oleh pengguna melalui notifikasi. Misalnya aplikasi pemutar musik [17].

2) *Background Service*

   Service yang melakukan operasi di *background* yang tidak akan terlihatoleh pengguna. Misalnya aplikasi yang mendeteksi atau membersihkan penyimpanan perangkat [17].

3) *Bound Service*

   Service yang akan terikat bila komponen aplikasi mengikatk dan memanggil fungsi *bindService()*. *Service* ini menawarkan tampilan antarmuka *client-server* yang memungkinkan komponen ini berinteraksi dengan *service* tersebut [17].

Service dapat memberikan keuntungan yang signifikan dalam pengembangan sebauh aplikasi *Android*. Aplikasi memungkinkan untuk melakukan tugas jangka panjang di *background* tanpa membebani pengguna. Hal yang harus diperhatikan dari *service* yaitu dalam penggunaanya karena dapat mengganggu kinerja perangkat dan menguras baterai [17].

2.8 *Wearable Device*

Wearable device atau perangkat yang dapat dikenakan merupakan sebuah perangkat yang merujuk pada teknologi yang dirancang untuk digunakan pada tubuh pengguna yang secara umumnya memberikan fungsionalitas yang terintegrasi dengan kehidupan sehari-hari seperti pemantauan aktivitas, kesehatan, pengaturan notifikasi, dan lain sebagainya.

Perangkat *wearable device* dapat berupa jam tangan pintar (*smartwatch*), gelang pintar (*smartband*), kaca mata pintar (*smart glasses*), dan perangkat lainnya yang sudah dilengkapi dengan teknologi-teknologi canggih seperti sensor giroskop, *Accelerometer*, GPS, sensor detak jantung, dan sensor-sensor lainnya yang dapat digunakan untuk pemantauan aktivitas dan kesehatan dari penggunanya.

Keunggulan yang diberikan lainnya yaitu dapat terintegrasi dengan *smartphone* dan juga menggabungkan kecerdasan buatan serta dapat terhubung dengan internet. Fitur-fitur canggih lainnya seperti dapat mendeteksi keadaan sekitar dan memperoleh berbagai informasi hingga dapat dimanfaatkan untuk mendeteksi jatuh pada penggunanya dengan menggunakan sensor *Accelerometer* yang diintegrasikan dengan mikrokontroler [18].

2.9 Visual Studio Code

Visual Studio Code merupakan kode editor yang ringan dan dapat dijalankan di berbagai sistem operasi seperti Windows, MacOs, dan Linux. Kode editor ini mendukung banyak bahasa pemrograman, *plug in, framework* dan *library*. Beberapa bahasa pemrograman diantaranya adalah JavaScript, C++, Python, Java, dan lain-lain. Kode editor ini digunakan oleh banyak pengembang karena ringantetapi *powerfull* untuk mengembangkan suatu aplikasi atau sistem dengan sifatnya yang *open source* sehingga dapat dikembangkan oleh siapa saja [19].

2.10 Kotlin

Kotlin merupakan salah satu pemrograman yang berjalan pada Java Virtual Machine (JVM) yang dirilis tahun 2011 dan dikembangkan oleh JetBrains. Kotlin mendapatkan popularitas dalam waktu singkat dengan fitur-fitur yang modern seperti null-safety, operation overloading, extension function, lambda expressions, dan lainnya yang membantu pengembang untuk menuliskan kode agar mudah dibaca dan lebih efisien [20].

Kotlin menjadi bahasa pemrograman utama untuk pengembangan *Android* yang diumumkan oleh Google secara resmi pada tahun 2017. Kotlin memiliki keuntungan interoperabilitas dengan Java yang di mana Kotlin dapat menggunakan kode Java yang sudah ada serta mempunyai syntax dan kemampuan untuk mengurangi jumlah kode yang ditulis [20].

2.11 JavaScript

JavaScript atau dikenal dengan singkatan JS, merupakan bahasa pemrograman *script* tingkat tinggi yang dikompilasi secara cepat *Just-in-time Compilation* (JIT). JavaScript adalah bahasa pemrograman yang memungkinkan untuk membuat halaman web interaktif dan merupakan bagian penting dari web. JavaScript merupakan bahasa Multi-paradigma yang mendukung gaya pemrograman berbasis fungsional dan imperatif [21].

JavaScript dapat digunakan untuk pengembangan aplikasi web baik *front-end* untuk tampilan *interface* maupun *back-end* untuk pengolahan data server dan *database*. Awalnya JavaScript dibuat untuk lingkungan browser dan membuat *website* menjadi interaktif. Namun, saat ini JavaScript dapat berjalan di luar browser seperti menggunakan Node.js [21].

2.12 NodeJs

NodeJs adalah *platform runtime* JavaScript yang dibangun untuk mengeksekusi kode JavaScript di luar browser. NodeJs telah banyak digunakan oleh perusahaan-perusahaan besar seperti Netflix, Paypal, eBay, dan lainnya. NodeJs menggunakan model pemrosesan *non-blocking* I/O yang memungkinkan untuk menangani banyak permintaan I/O secara bersamaan tanpa menghentikan proses permintaan yang lain. Selain itu, NodeJs mendukung pengembangan aplikasi secara *real-time* dan *streaming* [21].

NodeJs memiliki banyak modul yang sudah terpasang di dalamnya serta mengandung modul pihak ketiga yang dapat dipasang melalui *Node Package Manager* (NPM) yang dapat dengan mudah diintegrasikan dengan teknologi lain. NodeJs digunakan untuk pembangunan aplikasi web, server, *desktop*, *mobile*, dan lainnya [21].

2.13 ReactJs

ReactJs merupakan sebuah *framework* JavaScript yang digunakan untuk membangun u*ser interface* (UI) pada aplikasi web. ReactJs dikembangkan dan dirancang oleh Facebook yang bertujuan untuk mengembangkan aplikasi web dengan kompleksitas tinggi menjadi lebih mudah dengan kumpulan komponen-komponen yang dapat tersedia [22].

ReactJs menggunakan JSX (JavaScript XML) yang merupakan sebuah sintak untuk menggabungkan JavaScript dengan HTML. Dengan begitu, memungkinkan penulisan kode komponen yang lebih dekat dengan struktur tampilan antarmuka yang akan dikembangkan serta mempermudah pengembang dalam merancangnya [22].

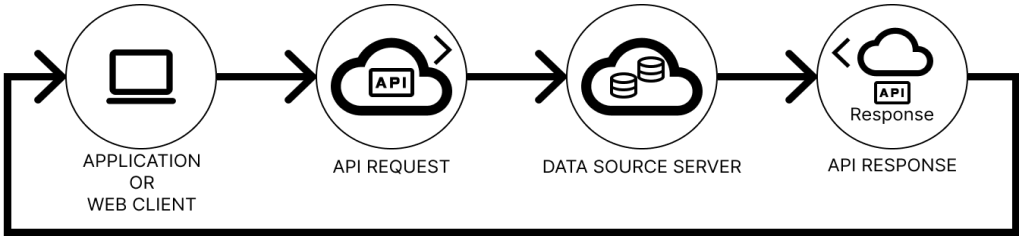
2.14 *Extensible Markup Language* (XML)

Extensible Markup Language atau disingkat XML merupakan bahasa *markup* yang sering digunakan untuk menampilkan data dalam bentuk teks yang terstruktur. XML biasanya digunakan saat pertukaran data antar aplikasi yang berbeda bahasa pemrograman serta *platform* yang digunakan.

XML digunakan di berbagai bidang seperti pada *web* servis, pengembangan *game*, pengembangan antarmuka aplikasi *mobile*, dan lain sebagainya. Struktur bahasa XML direpresentasikan dalam bentuk tag, atribut, dan nilai. Tag digunakan untuk menentukan jenis data yang digunakan, atribut digunakan untuk memberikan informasi terkait dengan data yang digunakan, dan nilai merupakan isi atau data itu sendiri.

2.15 *Application Programming Interface* (API)

Application Programming Interface atau disingkat API yang memungkinkan aplikasi atau web berkomunikasi dengan *server*. API memungkinkan pengembang untuk membuat aplikasi yang dapat menggunakan fungsi dari aplikasi lain atau *web* servis dengan cara mengintegrasikannya. Tujuannya adalah untuk saling bertukar data antar aplikasi dan mempercepat proses pengembangan aplikasi. *REST* API merupakan desain arsitektur yang terdapat di dalam API dan *RESTful* API merupakan REST *client* yang mengakses data pada *REST server* dari masing-masing sumber daya. Dengan tujuan dan cara kerja tersebut tentunya memudahkan pengembang dalam mengembangkan aplikasi karena pengembang dapat memakai fitur yang sudah ada tanpa perlu mengetahui detail bagaimana sumber daya itu diimplementasikan. Berikut merupakan alur cara kerja dari API [23] dapat dilihat pada Gambar 2.3.



Gambar 2.3 Cara Kerja API

Cara kerja API yaitu pertama aplikasi atau web *client* melakukan *request* untuk mengakses data pada sumber data di *server*. Lalu, *server* akan merespons permintaan dan mengembalikan data yang diminta yang biasanya berformat teks, JSON atau XML. Metode yang digunakan secara umum yaitu menggunakan metode HTTP yang dipakai dalam *REST* API seperti *GET*, *POST*, *PUT*, *DELETE*, dan *OPTIONS* [23].

Penjelasan metode HTTP yang dipakai dalam *REST API* adalah sebagai berikut:

1. *GET*, digunakan untuk mendapatkan data dari *REST server*.
2. *POST*, digunakan untuk membuat suatu data/sumber daya baru pada *REST server*.
3. PUT, digunakan untuk mengubah dan memperbaharui data/sumber daya pada *REST server*.
4. *DELETE*, digunakan untuk menghapus data/sumber daya pada *REST* API.
5. *OPTIONS*, digunakan untuk mendapatkan operasi yang didukung pada sumber daya dari *REST server*.

2.16 *JavaScript Object Notation* (JSON)

JavaScript Object Notation atau disingkat JSON merupakan format data yang mudah dibaca dan ditulis oleh manusia maupun mesin. Biasanya JSON digunakan sebagai format dalam pertukaran data antar aplikasi atau sistem yang menggunakan API. Struktur data pada JSON yaitu berbentuk objek (*object*) dan *array* [24].

Objek merupakan kumpulan pasangan informasi data berupa nama dan nilai ditulis dengan cara dibungkus dengan "{}" seperti contoh berikut "{nama: nilai}". Sedangkan *array* ditulis dengan cara dibungkus dengan "[]" dan pada setiap data

yang dikirimkan dipisahkan dengan tanda koma “,”. Tipe data JSON mendukung berbagai tipe data seperti *String*, *Boolean*, *Integer*, *null*, dan lain sebagainya. Keuntungan menggunakan JSON yaitu bahasa yang mudah dibaca oleh manusia dan komputer serta ringan sehingga proses atau operasi yang dijalankan dapat berjalan dengan cepat dan dapat memproses *gigabyte* data per detik pada satu inti prosesor yang umum [24].

2.17 Google Maps API

Google Maps API adalah sebuah platform pemetaan yang dikembangkan oleh Google. Fitur-fitur yang terdapat di dalam Google Maps API memungkinkan untuk diintegrasikan dengan aplikasi/*website* lain. Google Maps dapat menampilkan titik lokasi pengguna atau lokasi lain. Hal tersebut tentunya membantu pengembang aplikasi untuk mengembangkan aplikasi dengan fitur pemetaan seperti gambar-gambar tempat yang muncul pada aplikasi mereka. Gambar yang muncul pada aplikasi tersebut berasal dari hasil komunikasi antar aplikasi dengan *database* dari Google melalui API. Untuk menggunakan fitur-fitur tersebut diperlukan API *key* atau kunci API yang merupakan sebuah kode unik khusus untuk mengintegrasikan atau mengakses API tersebut dengan aplikasi/website tertentu agar dikenali oleh server Google Maps [25].

Penggunaan Google Maps API pada penelitian ini menggunakan akun dengan jenis *free tier* yang dimana mempunyai batasan permintaan untuk mengaksesnya. Untuk menggunakan layanan ini diperlukan akun yang terdaftar dan sudah melengkapi data akun tersebut beserta dengan kartu kredit. Berdasarkan dokumentasi resmi dari Google Maps Platform yang beralamat “https://mapsplatform.google.com/pricing”, akun jenis *free tier* mempunyai batasan permintaan $200 kredit atau setara dengan 28.500 *maploads* setiap bulan selama satu tahun tanpa biaya. Jika melakukan permintaan lebih dari itu maka akan dikenakan biaya sesuai harga dan permintaan yang dilakukan.

2.18 Google Directions API

Google Directions API merupakan sebuah layanan dari Google yang menyediakan informasi rute perjalanan yang dicari dari lokasi awal ke lokasi tujuan. Layanan ini juga memberikan informasi tentang jarak antara kedua lokasi,

waktu tempuh, dan informasi lainnya. Dengan adanya Google Directions API, maka dapat ditentukan rekomendasi rute menuju tempat tujuan berdasarkan faktor-faktor yang mempengaruhi dalam perjalanan seperti kemacetan, jarak tempuh, dan jenis kendaraan yang digunakan untuk menghasilkan rute yang optimal [26]. Cara kerja dari layanan ini yaitu dengan cara mengirimkan data *origin*, destinasi tempat, jenis kendaraan, dan data opsional lainnya. Lalu, Google Directions API akan memberikan respon berupa data dan informasi lengkap berdasarkan permintaan yang dikirimkan melalui aplikasi. Kemudian data tersebut dapat digunakan oleh aplikasi yang menggunakan layanan Google Directions API ini [27].

Penggunaan Google Directions API pada penelitian ini menggunakan akun dengan jenis *free tier* yang dimana mempunyai batasan permintaan untuk mengaksesnya. Untuk menggunakan layanan ini diperlukan akun yang terdaftar dan sudah melengkapi data akun tersebut beserta dengan kartu kredit. Berdasarkan dokumentasi resmi dari Google Maps Platform yang beralamat "https://mapsplatform.google.com/pricing", akun jenis *free tier* mempunyai batasan permintaan $200 kredit atau setara dengan 28.500 *maploads* setiap bulan selama satu tahun tanpa biaya. Jika melakukan permintaan lebih dari itu maka akan dikenakan biaya sesuai harga dan permintaan yang dilakukan. Layanan ini diidentifikasi melalui akun dan kunci unik (API *key*) untuk melakukan *request* atau permintaan.

2.19 Google Places API

Google Places API merupakan salah satu layanan dari Google yang menyediakan informasi terkait tempat-tempat di seluruh dunia seperti informasi toko, stasiun, terminal, tempat tertentu, dan lain-lain. Informasi yang diberikan berupa lokasi tempat, jarak tempuh, jam operasional, dan lainnya yang berhubungan dengan tempat tersebut. Cara kerja layanan ini yaitu aplikasi atau *website* mengirimkan data koordinat GPS (*latitude* dan *longitude*), radius untuk membatasi pencarian berdasarkan jarak agar lebih spesifik, kata pencarian yang ingin dicari, dan data opsional lainnya. Lalu, Google Places API akan memberikan respon berupa data dan informasi lengkap mengenai tempat-tempat yang dicari sesuai permintaan yang dikirimkan melalui aplikasi atau web [28].

Penggunaan Google Places API pada penelitian ini menggunakan akun dengan jenis *free tier* yang dimana mempunyai batasan permintaan untuk mengaksesnya. Untuk menggunakan layanan ini diperlukan akun yang terdaftar dan sudah melengkapi data akun tersebut beserta dengan kartu kredit. Berdasarkan dokumentasi resmi dari Google Maps Platform yang beralamat "https://mapsplatform.google.com/pricing", akun jenis *free tier* mempunyai batasan permintaan $200 kredit atau setara dengan 28.500 *maploads* setiap bulan selama satu tahun tanpa biaya. Jika melakukan permintaan lebih dari itu maka akan dikenakan biaya sesuai harga dan permintaan yang dilakukan. Layanan ini diidentifikasi melalui akun dan kunci unik (API *key*) untuk melakukan *request* atau permintaan.

2.20 OpenWeatherAPI

OpenWeatherAPI adalah layanan web yang menyediakan berbagai informasi cuaca di seluruh dunia baik secara *real-time*, histori, atau prediksi cuaca beberapa hari ke depan. Informasi yang diberikan berupa kondisi cuaca, suhu, kelembaban, tekanan udara, kecepatan angin, dan lain-lain. Format data yang didukung pada layanan ini yaitu JSON dan XML, dan HTML. Layanan API ini dapat diintegrasikan dengan aplikasi *mobile*, web, dan *desktop*. Cara kerja pada layanan OpenWeatherAPI yaitu aplikasi yang terintegrasi mengirimkan permintaan berupa lokasi, data yang dibutuhkan, dan data opsional lainnya seperti bahasa, limit, dan jenis atau format data. Lalu, layanan ini akan memberikan respon yang diminta berupa data lengkap mengenai cuaca [29].

Penggunaan API dari OpenWeatherAPI pada penelitian ini yaitu menggunakan jenis akun *free*. Berdasarkan dokumentasi dari OpenWeatherAPI yang beralamat "https://openweathermap.org/price", penggunaan API diidentifikasi melalui kunci unik (API *key*) dari setiap akun. API dengan jenis akun *free* ini memiliki batasan 60 permintaan/menit dan 1 juta permintaan/bulan dan hanya bisa menggunakan beberapa layanan yang disediakan. Untuk menggunakan layanan lainnya serta menambah batasan permintaan di setiap bulannya, dapat berlangganan secara langsung dan memilih jenis akun sesuai dengan kebutuhan.

2.21 Amazon *Web Service* (AWS)

Amazon *Web Service* (AWS) merupakan sebuah *platform* komputasi luas yang menawarkan berbagai layanan untuk pembangunan aplikasi yang dapat di kustomisasi sesuai dengan kebutuhan. AWS bertanggung jawab dalam menangani skalabilitas, performa, keamanan, dan kemudahan dalam penerapannya [30].

2.21.1 Amazon *Elastic Compute Cloud* (EC2)

Amazon *Elastic Compute Cloud* (EC2) merupakan salah satu layanan komputasi awan dari Amazon. Layanan yang diberikan yaitu penyewaan *instance* vitrual pada infrastruktur komputasi AWS. Pengguna layanan dapat mengatur kebutuhan mereka sendiri seperti mengatur memori, penyimpanan, dan *Central Processing Unit* (CPU). AWS memberikan kebebasan terhadap penggunanya seperti dapat membuat lebih dari satu *instance* untuk keperluan aplikasi yang dibangun. AWS EC2 juga menyediakan berbagai fitur yang tentunya membantu dalam pengelolaan server seperti *monitoring, auto scaling* yang menyesuaikan *instance* bertambah atau berkurang sesuai dengan permintaan yang masuk dari pengguna plikasi, dan lainnya [30].

Penggunaan EC2 pada penelitian ini yaitu menggunakan akun jenis *free tier* yang dimana memiliki batasan dalam mengaksesnya. Untuk mengakses dan menggunakan layanan ini diperlukan akun yang sudah terdaftar beserta dengan kartu kredit. Berdasarkan dokumentasi resmi dari AWS yang beralamat "https://aws.amazon.com/free", akun dengan jenis *free tier* memiliki akses gratis selama 12 bulan sejak mendaftar dengan batasan 750 jam/bulan dengan *instance* tertentu sesuai daerah. Waktu tersebut dihitung dari setiap *instance* yang digunakan. Jika melebihi dari batas yang diberikan maka akan dikenakan biaya sesuai penggunaan.

2.21.2 Amazon *Relational Database Service* (RDS)

Amazon *Relation Database Service* (RDS) merupakan salah satu layanan dari Amazon yang dapat mengelola *database* dengan mengoperasikan dan sakalabilitas relasi *database* di *cloud.* Layanan ini membantu pengembang aplikasi karena mereka dapat fokus untuk mengembangkan aplikasi dan menyimpan data ke dalam RDS karena RDS bertanggung jawab dalam mengelola server termasuk

keamanan data. RDS juga mendukung berbagai jenis *database* seperti MySQL, MariaDB, Oracle, dan lainnya [30].

Penggunaan RDS pada penelitian ini yaitu menggunakan akun jenis *free tier* yang dimana memiliki batasan dalam mengaksesnya. Untuk mengakses dan menggunakan layanan ini diperlukan akun yang sudah terdaftar beserta dengan kartu kredit. Berdasarkan dokumentasi resmi dari AWS yang beralamat "https://aws.amazon.com/free", akun dengan jenis *free tier* memiliki akses gratis selama 12 bulan sejak mendaftar dengan batasan 750 jam/bulan dan 20GB peenyimpanan *database* menggunakann *Solid State Drive* (SSD) serta 20GB cadangan penyimpanan. Jika melebihi dari batas yang diberikan maka akan dikenakan biaya sesuai penggunaan.

2.22 *Web Service*

Web service adalah layanan web yang digunakan untuk berkomunikasi satu sama lain antara perangkat elektronik satu ke perangkat elektronik lain melalui *World Wide Web* (WWW) atau server yang berjalan pada komputer, menerima permintaan di *port* tertentu melalui jaringan yang melayani dokumen web seperti HTML, JSON, XML, dan sebagainya, lalu membuat layanan web yang berfungsi untuk menyelesaikan masalah domain tertentu (WWW, HTTP).

Salah satu cara untuk melakukan permintaan tersebut yaitu dengan memasukkan informasi ke dalam metode HTTP yang di mana memiliki lima metode yang umum digunakan seperti GET, HEAD, PUT, DELETE, dan POST. Tentunya HTTP sudah ter standarisasi meskipun beberapa *web service* memilih untuk menggunakan nama metode yang lebih spesifik untuk aplikasi seperti menggunakan jalur URI atau dokumen permintaan [31].

2.23 XAMPP

XAMPP adalah sebuah aplikasi yang biasa digunakan oleh pengembang untuk mengembangkan, mengelola, dan membangun sebuah web *server* secara lokal. XAMPP memiliki beberapa fitur seperti Apache HTTP *server,* MySQL, MariaDB *database*, dan penerjemah bahasa *script*. Berikut penjelasan secara singkat mengenai beberapa fitur yang tersedia di XAMPP:

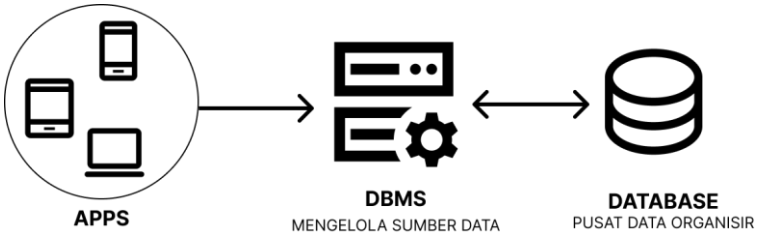
1. Apache HTTP server, digunakan sebagai *server* web utama yang bersifat *open source*.
2. MariaDB *database*, digunakan untuk manajemen *database* dan bersifat *open source*, MariaDB juga merupakan cabang dari MySQL.
3. Mendukung beberapa bahasa *script* seperti Perl, PHP, dan Python yang digunakan untuk membangun aplikasi web.

XAMPP mempunyai kompatibilitas terhadap berbagai sistem operasi seperti Windows, Linux, dan MacOS. XAMPP menjadi teknologi yang digunakan dalam implementasi web *service* untuk memperbaiki performa bisnis. Terutama jika XAMPP digunakan pada sistem operasi Linux Ubuntu, berdasarkan penelitian menunjukkan bahwa XAMPP mempunyai performa lebih baik pada kecepatan dan keandalan dalam merespons [32].

2.24 Basis Data MySQL (*Database*)

Basis dapat diartikan sebagai tempat yang digunakan untuk mengumpulkan dan menyimpan barang atau benda. Data merupakan representasi fakta pada dunia nyata yang dapat berupa objek seperti manusia, benda, dan lainnya yang disimpan dalam bentuk teks, angka, bunyi gambar, simbol, atau kombinasi dari semuanya. Jadi, basis data merupakan kumpulan data yang saling berhubungan satu sama lain yang diorganisasikan berdasarkan skema atau struktur tertentu, kemudian disimpan di dalam perangkat keras komputer dan digunakan dengan perangkat lunak untuk mengoperasikannya seperti memperbaharui data, mencari data, menambah data, dan menghapus data dengan tujuan tertentu [33].

MySQL adalah sebuah sistem pengorganisasian dan pengolahan *database* pada komputer atau bisa disebut *Databse Management System* (DBMS). Berikutnya yaitu *Relation Database Management System* (RDBMS) yang sama dengan DBMS dan mengorganisasikan data dalam suatu struktur serta menghubungkan bais data yang disimpan. MySQL itu sendiri merupakan *Structured Query Language* (SQL) yang merupakan perangkat lunak bersifat *open source* yang digunakan oleh banyak pengembang untuk membangun aplikasi yang berbasis *databse* [34]. Berikut merupakan gambaran dari *database* dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Database Management System* (DBMS)

2.25 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah standar enkripsi yang dikembangkan oleh Joan Daemen dan Vincent Rijmen. AES merupakan algoritma simetris yang menyimpan data dengan *key* yang sama saat melakukan proses enkripsi dan dekripsi. Terdapat beberapa jenis *key* AES antara lain yaitu AES-128, AES-192, dan AES-256. AES mendukung 128-bit hingga 256-bit *key-length* yang tahan terhadap berbagai serangan seperti serangan *exhausting key search* karena memiliki panjang kunci setidaknya 128 bit yang terdapat 3.14 x 10^{38} kemungkinan yang dimana dibutuhkan waktu cukup lama untuk menemukan *key* tersebut [35]. Skala Likert

2.26 Git

Git merupakan sebuah sistem *version control* yang dirancang untuk menangani proyek kecil hingga proyek yang sangat besar dengan efisiensi dan kecepatan yang cepat. Git dilengkapi dengan fitur *branching* lokal atau cabang dalam lingkup pengembangan perangkat lunak. *Branching* berjalan secara independen yang memungkinkan pengembang mencoba sebuah ide atau fitur baru tanpa khawatir dengan error yang akan terjadi pada aplikasi di *branch* utama. Fitur lainnya yaitu *merging* dimana berfungsi untuk menggabungkan satu *branch* dengan *branch* lainnya. Git sangat membantu pengembang untuk memantau kode yang ditulis agar lebih mudah dalam memberikan versi dari kode yang dikembangkan [36].

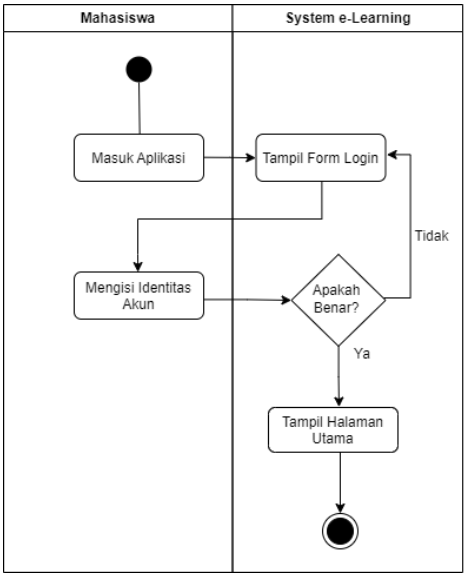
2.27 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa untuk memvisualisasikan, menspesifikasikan, memodelkan, dan membangun dengan cara mendokumentasikan sistem perangkat lunak berorientasi objek [37]. UML

memiliki berbagai diagram, empat diagram diantaranya yang akan digunakan pada penelitian ini yaitu *Activity Diagram, Use Case Diagram, Class Diagram, dan Sequence Diagram* [38].

2.27.1 *Activity* Diagram

Activity diagram merupakan notasi yang sering digunakan untuk merepresentasikan grafik dari aktivitas pada sistem yang berorientasi objek. Aktivitas diagram memiliki beberapa simbol khusus untuk penggambarannya yang pada setiap simbolnya memiliki fungsinya masing-masing. Simbol yang digunakan bertujuan untuk memudahkan dalam memodelkan aktivitas [39]. Berikut contoh dari aktivitas diagram dapat dilihat pada Gambar 2.5.



Gambar 2.5 Contoh *Activity* Diagram

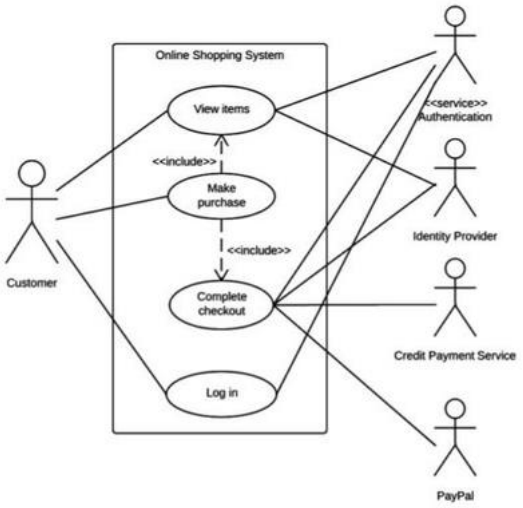
Pada contoh gambar di atas menggambarkan proses dari mahasiswa yang melakukan *log in* ke dalam aplikasi *System e-Learning* yang dimulai dari mahasiswa yang masuk ke dalam aplikasi, lalu muncul halaman log in. Mahasiswa kemudian mengisikan identitas akun dan dilanjutkan dengan proses validasi akun. Jika benar, maka mahasiswa akan diarahkan langsung ke halaman utama. Jika salah, mahasiswa akan dikembalikan ke halaman *log in* kembali. Berikut penjelasan dari setiap simbol pada aktivitas diagram di atas:

1. *Swimlane* (Berbentuk seperti tabel), memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.

2. *Initial Node* atau status awal (Lingkaran penuh), objek dibentuk dan dimulainya suatu proses pada aktivitas.
3. *Activity* atau aktivitas (Persegi panjang), Menampilkan pekerjaan atau proses yang dilakukan di setiap *swimlane*.
4. *Decision* atau Percabangan (Belah ketupat), percabangan yang di mana terdapat pilihan aktivitas lebih dari satu.
5. *Activity Final Node* atau status akhir (Lingkaran yang terdapat cincin), objek dibentuk yang menandakan akhir dari proses aktivitas.

2.27.2 *Use Case* Diagram

Use case merupakan representasi dari interaksi yang melibatkan pengguna dengan sistem yang menunjukkan hubungan atau keterkaitan. Tujuan dar *use case* sendiri yaitu untuk mendefinisikan perilaku koheren tanpa mengungkap struktur internal dari suatu sistem. Dalam penggambaran diagram *use case* terbagi menjadi dua item yaitu aktor dan skenario [39]. Aktor adalah orang atau sistem, atau organisasi yang berinteraksi dengan sistem yang dibuat. Penggambaran aktor yaitu digambar dalam diagram sebagai lingkaran luar. Sedangkan skenario adalah urutan interaksi antar aktor dan sistem. Skenario digambarkan sebagai alur dari setiap *use case* yang mencakup serangkaian tindakan yang diambil oleh aktor dan sistem sebagai respons. Berikut contoh gambar dari diagram *use case* yang menunjukkan interaksi dari setiap aktor dengan sistem belanja *online*. Berikut merupakan contoh dari *Use Case* Diagram dapat dilihat pada Gambar 2.6



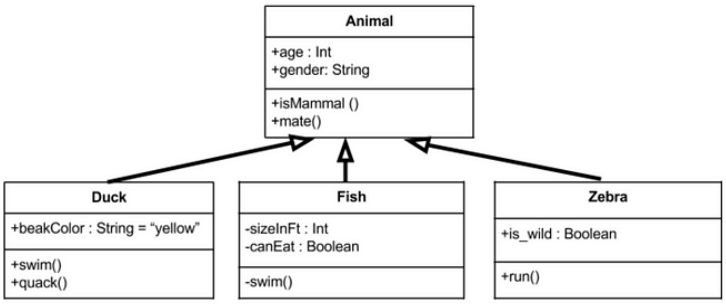
Sumber Gambar: https://www.detik.com/bali/berita/d-6502555/use-case-diagram-simbol-komponen-cara-membuat-dan-contoh

Gambar 2.6 Contoh *Use Case* Diagram

2.27.3 *Class* Diagram

Class diagram digunakan untuk merepresentasikan struktur dari suatu sistem berbasis objek. Class diagram menyajikan kelas-kelas yang terdapat pada sistem serta relasi dari setiap *calss* tersebut. Tujuan dari *class* diagram yaitu untuk membantu komunikasi dalam memahami struktur dari suatu sistem yang dibuat atau dikembangkan [39].

Class diagram mendeskripsikan setiap *class*, *package*, metode, dan objek. *Class* diagram juga menggambarkan hubungan atau *relation* antar *class* seperti pewarisan atribut, asosiasi, dan lainnya [39]. Berikut contoh sederhana dari *class* diagram dapat dilihat pada Gambar 2.7.



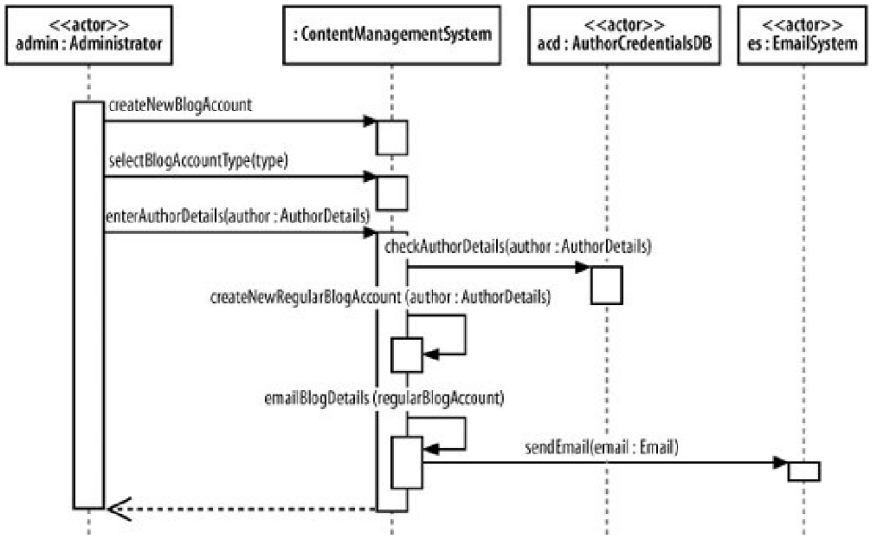
Sumber Gambar: https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b

Gambar 2.7 Contoh *Class* Diagram

Dari contoh *class* diagram di atas, terlihat ada empat *class* yang masing-masing memiliki atribut (baris kedua), operasi (baris ketiga), dan pewarisannya. Ditunjukkan bahwa *parent class* (Animal) yang mewarisi semua anggota dari setiap *class Duck, Fish*, Zebra.

2.27.4 *Sequence* Diagram

Sequence diagram menunjukkan urutan dari sebuah *use case* yang bermaksud untuk memodelkan proses bisnis atau teknis dalam sebuah sistem yang dapat membantu pengembang untuk memahami urutan aktivitas yang terjadi. kotak vertikal merepresentasikan objek-objek dan urutan pesan antara objek direpresentasikan oleh panah horizontal [39]. Berikut merupakan contoh dari *sequence* diagram yang menggambarkan mahasiswa melakukan *log in* dapat dilihat pada Gambar 2.8.



Gambar 2.8 Contoh *Sequence* Diagram

Pada gambar 2.8 tersebut, terdapat satu aktor dan tiga objek, yaitu Administrator sebagai aktor dan objek *ContentManagementSystem*, objek *AuthorCredentialsDB*, dan objek *EmailSystem*. Proses bisnis yang terjadi yaitu Admin membuat akun billing baru, memilih tipe akun blog, dan memasukan detail *Author* yang kemudian dilakukan dikelola pada objek *ContentManagementSystem* lau dilakukan pengecekan pada objek *AuthorCredentialsDB*. Lalu kemudian mengirimkan email pada objek *EmailSystem* yang juga dikembalikan ke Administrator [38].

2.28 Skala Likert

Skala Likert merupakan sebuah teknik yang memungkinkan responden untuk menilai suatu item pada lima hingga tujuh poin tergantung pada jumlah perjanjian atau ketidaksepakatan mereka pada item tersebut. Terdapat dua bagian pada Skala Likert yakni bagian item dan evaluasi. Bagian item biasanya berupa pernyataan tentang suatu produk, acara, atau lainnya. Sedangkan bagian evaluasi merupakan daftar dari tanggapan responden seperti “Sangat Setuju” hingga “Sangat Tidak Setuju” [40]. Hasil dari penilaian responden tersebut diakumulasikan lalu dihitung untuk mendapatkan kesimpulan pengujian. Jumlah sampel yang disarankan yaitu minimal sebanyak 30 responden [41].