BAB 2
TINJAUAN PUSTAKA

2.1 Yayasan Masjid Besar At-Taqwa Gegerkalong KPAD

Yayasan At – Taqwa (YAT) merupakan induk organisasi atau organisasi tertinggi di dalam lingkungan yang berada di daerah Masjid At Taqwa KPAD Gegerkalong. YAT didirikan memiliki fungsi untuk mengelola dan memelihara banyak unit yang berada di bawahnya, seperti: Dewan Kemakmuran Masjid (DKM), Unit Pengelola Zakat (UPZ), Pendidikan Al Quran (PAQ), TK Islam Terpadu (TKIT), Sekolah Dasar Islam Terpadu (SDIT) dan Sekolah Tinggi Ilmu Tarbiyah (STIT). Kegiatan-kegiatan tersebut mencakup berbagai bidang sesuai dengan fungsi-fungsi ibadah, sosial dan kegiatan muamalah, pendidikan dan kebudayaan. Keseluruhan kegiatan yang dilaksanakan dari tahun ke tahun, menunjukkan adanya upaya peningkatan dalam berbagai bentuk yang bervariasi, namun senantiasa merujuk kepada pola pembinaan yang terarah.

2.1.1 Sejarah Organisasi

Perwujudan pembangunan, pengembangan, dan peningkatan peranan masjid beserta seluruh aspek kegiatannya, perlu diimbangi serta diiringi dengan dukungan perangkat organisasi dan manajemen yang efektif. YAT harus dapat menjadi payung hukum untuk seluruh kegiatan yang dijalankan. Untuk itu, pada tanggal 19 Mei 1996 bertepatan dengan tanggal 1 Muharram 1417 Hijriyah, diresmikanlah berdirinya suatu Yayasan yang diberi nama Yayasan “AT-TAQWA” Komplek Perumahan Angkatan Darat berdasarkan Akta Notaris No. 109 A Tanggal 22 November 1996. Yayasan “AT-TAQWA” berkedudukan di Masjid At-Taqwa, organisasi kepengurusan yang ditetapkan 5 tahun sekali.

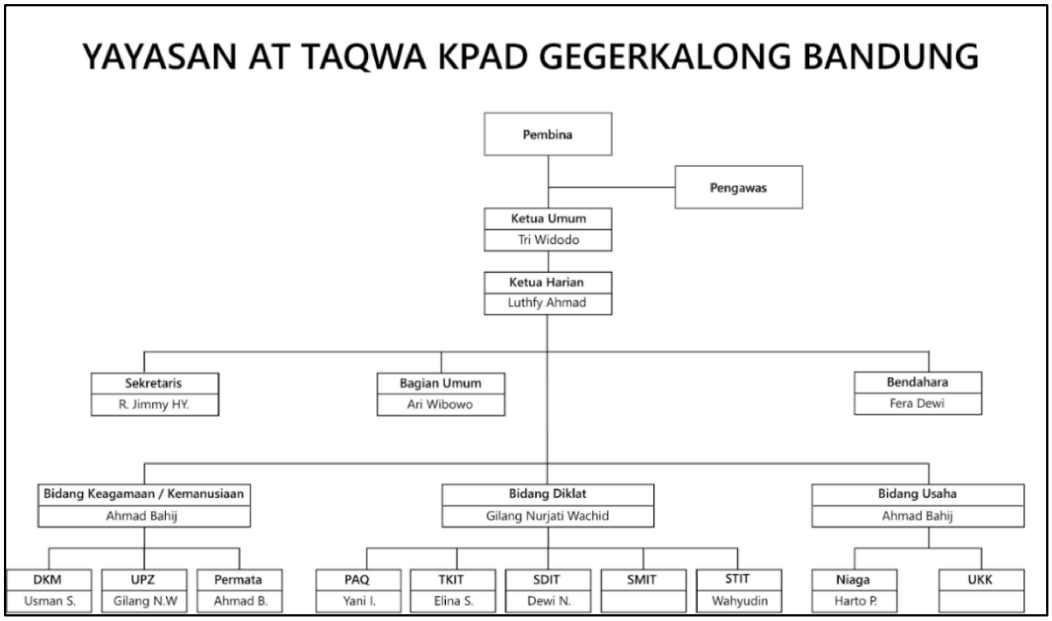
2.1.2 Visi dan Misi

Adapun kegiatan yang saat ini dikelola Yayasan At-Taqwa meliputi bidang Keagamaan, Pendidikan dan Pelatihan, serta Kemanusiaan-Sosial-Usaha. Setiap bidang memiliki Unit-unit Kerja dengan visi misi yang selaras dengan visi misi masjid dan Yayasan At-Taqwa.

1. Membina dan memupuk saling pengertian serta memelihara kerja sama yang baik di antara para pembina, pengurus dan jamaah masjid.
2. Mengembangkan serta meningkatkan mutu dan martabat serta kemakmuran masjid pada umumnya secara gotong-royong (Taawun) dan tasamuh dalam membina ukhuwah Islamiyah, Wahidatul Ummah dan Akhlak Karimah kaum muslimin berdasarkan akidah Islamiyah serta senantiasa berpedoman kepada ajaran Islam.
3. Meningkatkan serta mengembangkan syiar Islam dalam berbagai aspek kehidupan dengan menggunakan sarana ilmu pengetahuan, seni budaya, teknologi, sosial, ekonomi dan sebagainya.
4. Meningkatkan mutu dan kualitas Iman, Islam dan Ihsan dalam penjabaran dan pengalamannya.
5. Menegakkan, mengisi dan mengamalkan dasar negara Pancasila.

2.1.3 Struktur Organisasi

Berdasarkan hasil wawancara dengan Pengurus Yayasan Masjid Besar At-Taqwa Gegerkalong KPAD, berikut adalah struktur organisasi YAT:



Gambar 2. 1 Struktur Organisasi Yayasan Masjid At Taqwa KPAD Gegerkalong

2.1.4 Rancang dan Bangun

Perancangan aplikasi meliputi proses identifikasi kebutuhan pengguna berupa kebutuhan fungsional dan non-fungsional agar dapat memenuhi tujuan pengguna dalam menggunakan aplikasi. Pada tahapan ini, biasanya individu atau tim merancang berbagai faktor seperti sistem operasi, pemilihan bahasa dan atau kerangka kerja perangkat lunak yang akan digunakan, desain antarmuka, struktur *database,* algoritma, pemodelan kebutuhan dan faktor lainnya. Tujuan utama perancangan adalah menentukan tujuan dan langkah apa yang perlu dilakukan untuk memenuhi kebutuhan pengguna.[2]

Ketika tahap perancangan telah selesai, maka yang perlu dilakukan selanjutnya adalah pembangunan aplikasi yang melibatkan pengembangan, implementasi perancangan ke dalam tahap kode, *testing*, *debugging* dan *maintenance*. Tahapan ini memerlukan keahlian khusus dalam teknis agar memastikan bahwa aplikasi yang dikembangkan dapat memenuhi sesuai *timeline*, standar kualitas dan kebutuhan pengguna. Perancangan dan pembangunan aplikasi ini menggunakan pendekatan *Waterfall* model yang bersifat sistematik berdasarkan persyaratan proyek telah ditetapkan dengan jelas dan tidak mungkin berubah secara signifikan. [4]

2.2 Sistem Presensi Digital

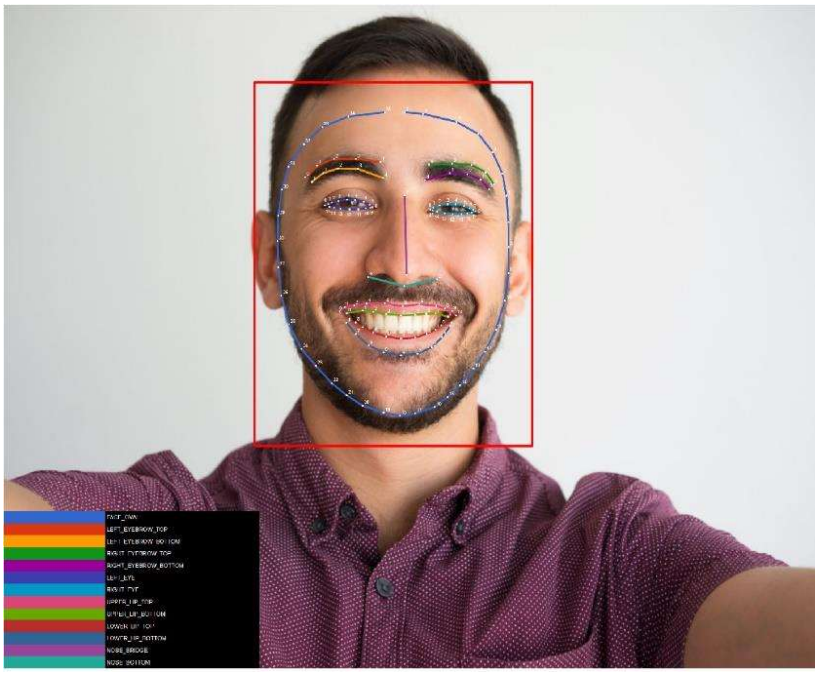
Sistem presensi digital berupa sistem perangkat lunak yang digunakan untuk mencatat absensi atau kehadiran pengguna menggantikan sistem tradisional menggunakan teknologi digital. Dalam penulisan ini, sistem presensi digital dilakukan melalui perangkat lunak yang dapat diakses melalui perangkat *smartphone*.[5]

Untuk merekam kehadiran pegawai secara digital dimanfaatkan teknologi pengenalan wajah (*Face Recognition)* dan lokasi geografis *(Geofence)*. Teknologi *Face Recognition* akan memproses data wajah yang ditangkap oleh kamera melalui *smartphone* dan membandingkan dengan data wajah yang telah terdaftar pada *database*. Disisi lain, sistem menggunakan teknologi *geofence* untuk memverifikasi lokasi pengguna dalam suatu batasan virtual geografis yang telah ditentukan.[6]

Sehingga, kedua teknologi tersebut dapat meningkatkan keabsahan *input* kehadiran sekaligus meminimalkan peluang terjadinya kecurangan absensi.

2.3 Face Recognition

Face recognition adalah bagian dari masalah sub-divisi pengenalan pola visual. Manusia secara terus-menerus mengenali pola visual dan mendapatkan informasi visual melalui mata. Bagi komputer, baik itu gambar atau video digambarkan sebagai matriks dari banyak *pixel*. *Face recognition,* mampu mengidentifikasi identitas karakteristik wajah yang unik dan khas, seperti bentuk wajah, jarak antara mata, hidung, dan mulut, serta pola iris (*visual patterns)*.[7]



Sumber gambar:
https://developers.google.com/ml-kit/vision

Gambar 2. 2 Titik Penting pada Face Recognition

Teknologi pengenalan wajah telah banyak digunakan dalam bidang keamanan karena kemudahannya dan keakuratannya. Berikut adalah beberapa poin penjelasan dari alur kerja dari *face recognition* :

1. Face and Landmark Localization

Pendeteksian wajah memisahkan area wajah dari latar belakang. Pada kasus ini wajah yang terdeteksi mungkin perlu dilacak di beberapa *frame* menggunakan

komponen pelacakan wajah. Sementara pendeteksian wajah memberikan perkiraan kasar lokasi dan skala wajah, mengidentifikasi dan memberikan penanda titik penting (*landmark)* pada wajah seperti mata, hidung, mulut, dan garis wajah. [8]

2. Face Normalization

Tahapan ini dilakukan untuk memperoleh normalisasi geometri dan fotometri pada wajah. Hal ini diperlukan karena diharapkan mampu mengenali gambar wajah dengan variasi pose dan pencahayaan. Proses normalisasi geometri mengubah wajah menjadi bingkai standar dengan cara memotong wajah. Proses normalisasi fotometri menghasilkan normalisasi wajah berdasarkan sifat seperti pencahayaan dan skala abu-abu. [8]

3. Feature Extraction

Ekstraksi gambar pada wajah dilakukan pada wajah yang telah dinormalisasi untuk mengekstrak informasi penting untuk membedakan wajah dari orang yang berbeda terhadap variasi geometri dan fotometri. Fitur wajah yang diekstrak digunakan untuk tahapan selanjutnya pencocokan wajah. [8]

4. Feature Matching

Dalam pencocokan wajah, fitur yang diekstrak dari wajah masukan dicocokkan dengan salah satu atau banyak wajah terdaftar dalam *Database of Enrolled users*. *Matcher* mengeluarkan *output* “ya” atau “tidak” untuk verifikasi 1:1; untuk identifikasi 1:N, keluarannya adalah identitas dari wajah masukan saat pencocokan terbaik ditemukan dengan nilai keyakinan (*confidence*) yang cukup atau tidak dikenal saat skor pencocokan terbaik berada di bawah ambang batas. Tantangan utama dalam tahap pengenalan wajah ini adalah untuk menemukan metrik kesamaan yang cocok untuk membandingkan fitur wajah. [8]

5. Database of Enrolled Users

Database of enrolled users merupakan *database* yang berisi informasi data pengguna yang telah terdaftar dan diizinkan untuk diakses oleh sistem *face recognition*. Informasi yang disimpan dalam *database* ini meliputi data wajah, dan

proses ini terjadi ketika *Feature Matching* membutuhkan data wajah dalam mencocokkan wajah masukan dan wajah yang telah terdaftar. [8]

2.4 Google Machine Learning Kit

Google ML Kit adalah *Software Development Kit* menghadirkan keahlian *machine learning Google* ke aplikasi Android dan iOS dalam paket yang kuat dan mudah digunakan. ML Kit menyediakan API praktis yang dapat membantu developer menggunakan model *TensorFlow Lite* terkustomisasi di aplikasi *mobile*. Pemrosesan ML Kit terjadi di perangkat *mobile*, sehingga membuatnya cepat dan dapat memungkinkan penggunaan *real-time* seperti pemrosesan *input* kamera. ML Kit ini juga berfungsi saat *offline* dan dapat pula digunakan untuk memproses gambar dan teks yang harus tetap ada di perangkat. [9]

Google ML Kit memfasilitasi tempat penyimpanan data dan penyajian model *machine learning* pada perangkat *mobile*. Hal ini dapat mengurangi beban tugas menjalankan model *machine learning* pada perangkat *mobile* terhadap panggilan API yang mencakup kasus penggunaan *mobile* umum seperti face *detection, text recognition, barcode scanning, image labeling,* dan *landmark recognition.* Proses ini hanya membutuhkan *input* sebagai parameter untuk menghasilkan banyak informasi analitis. API yang disediakan oleh ML Kit dapat berjalan di perangkat, *cloud*, maupun keduanya. API pada perangkat tidak bergantung pada koneksi jaringan sehingga dapat bekerja lebih cepat dibandingkan dengan API berbasis *cloud*. API berbasis *cloud* ditampung di *Google Cloud* Platform dan menggunakan teknologi *machine learning* untuk memberikan tingkat akurasi yang lebih tinggi. Jika API yang tersedia tidak mencakup kasus penggunaan yang diperlukan, model *TensorFlow Lite* terkustomisasi dapat dibuat sampai disajikan menggunakan *Firebase*. ML Kit bertindak sebagai lapisan API di antara model-model yang terkustomisasi sehingga membuatnya mudah dijalankan.[9]

2.5 Tensorflow

TensorFlow merupakan platform *open source* untuk *machine learning*. Platform ini memiliki ekosistem alat, perpustakaan, dan sumber daya komunitas

yang komprehensif dan fleksibel sehingga memungkinkan peneliti untuk mendorong kecanggihan *machine learning* dan pengembang untuk dengan mudah membangun dan menerapkan aplikasi yang didukung *machine learning*. *TensorFlow* beroperasi dalam skala besar dan di lingkungan yang heterogen. Model komputasinya didasarkan pada grafik aliran data dengan status yang dapat diubah. *Node* grafik dapat dipetakan ke mesin yang berbeda dalam sebuah *cluster*, dan dalam setiap mesin ke CPU, GPU, dan perangkat lainnya. Platform ini berfungsi sebagai platform untuk penelitian dan penerapan sistem *machine learning* di banyak bidang, seperti *speech recognition, computer vision, robotics, information retrieval,* dan *natural language processing* [10]. *TensorFlow* awalnya dibuat oleh peneliti di Google dan merupakan salah satu yang paling populer di antara *deep learning libraries*. Di bidang *deep learning, neural networks* telah mencapai kesuksesan luar biasa dan implementasinya telah mendapatkan popularitas di berbagai bidang lain. *TensorFlow* di sini memudahkan dan mempercepat penelitian dan penerapan model neural *networks* [11].

TensorFlow Lite adalah versi ringan dari *TensorFlow* yang membantu penerapan model *machine learning* di perangkat Android dan iOS. Versi ini memanfaatkan kekuatan Android Neural Network API untuk mendukung akselerasi perangkat keras. Konversi model *TensorFlow* menjadi *TensorFlow Lite* diperlukan sebelum kita dapat menggunakannya di perangkat *mobile*. Hal ini penting karena model *TensorFlow* lebih besar dan mengalami lebih banyak latensi daripada model *Lite* yang sudah dioptimalkan untuk berjalan di perangkat *mobile*[9].

2.6 MobileFaceNet

MobileFaceNet merupakan sebuah model pembelajaran jaringan saraf konvolusional atau istilah dalam bahasa asing adalah *Convolutional Neural Network* (CNN) yang sering digunakan khusus untuk mendeteksi objek, dan model ini dapat digunakan pada perangkat *mobile* yang memiliki pemrosesan daya komputasi dan memori yang terbatas. CNN dirancang untuk pengolahan *data grid* seperti gambar, arsitektur CNN menyerupai sistem saraf manusia dan memberikan

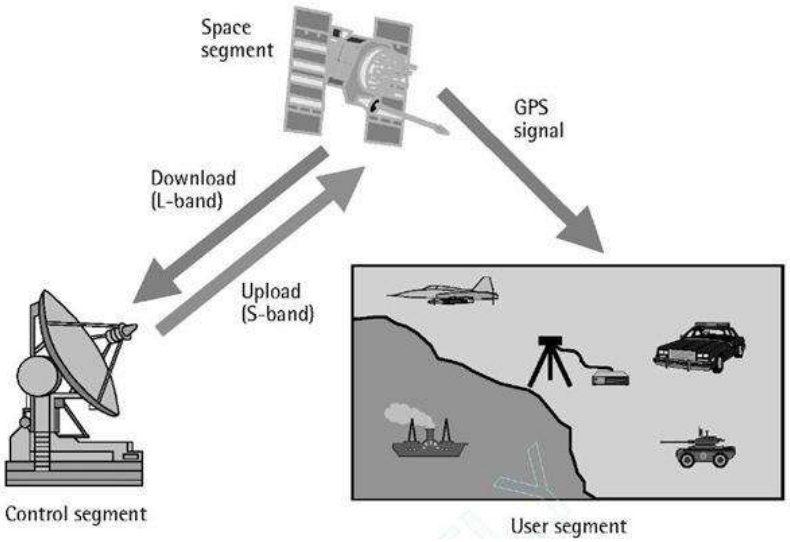
manfaat untuk pengenalan pola visual termasuk verifikasi wajah, di mana sering dipergunakan untuk teknologi *authentication* di aplikasi *mobile* seperti membuka kunci layar, *login* aplikasi, pembayaran secara *mobile.*[12]

MobileFaceNet memiliki arsitektur jaringan saraf tiruan yang didasarkan pada *MobileNetV2* dengan modifikasi yang mencakup penyesuaian struktur jaringan, penggantian fungsi aktivasi, dan pengenalan modul perhatian saluran yang efektif (ECA). *MobileFaceNet* menggunakan lapisan konvolusi untuk mengekstraksi fitur-fitur wajah, blok *bottleneck* untuk mengurangi kompleksitas model, lapisan *pooling* untuk mengurangi dimensi fitur, dan lapisan *fully connected* untuk klasifikasi wajah. Untuk meningkatkan akurasi, *MobileFaceNet* menggantikan fungsi aktivas dengan *Rectified Linear Unit* (*ReLU*) dan memperkenalkan modul perhatian saluran yang efektif (ECA) untuk meningkatkan representasi fitur.[13], [14]

Model pembelajaran jaringan saraf tiruan ini dapat diintegrasikan ke dalam aplikasi *mobile*, memungkinkan verifikasi wajah untuk proses presensi dapat dilakukan melalui perangkat *mobile*.

2.7 Global Positioning System

GPS (*Global Positioning System*) adalah sistem navigasi satelit yang digunakan untuk menentukan lokasi secara akurat. GPS digunakan dalam berbagai aplikasi, termasuk navigasi personal, navigasi penerbangan, navigasi kendaraan, navigasi laut, pemetaan (*mapping*), survei, dan infrastruktur. Dewasa ini, banyak perangkat elektronik seperti *smartphone*, laptop, dan tablet telah dilengkapi dengan GPS, sehingga dapat menentukan posisi mereka dengan akurasi yang tinggi di mana pun di dunia. Namun, sistem GPS juga memiliki beberapa keterbatasan, seperti lingkungan tertentu dalam lingkungan bawah tanah. [15]



Sumber gambar:
Buku *Introduction to GPS: The Global Positioning System*

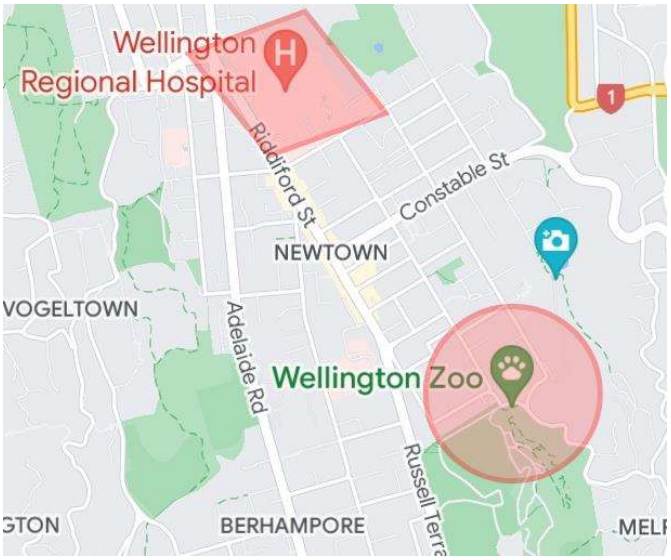
Gambar 2. 3 Cara bekerja GPS

GPS terdiri dari tiga komponen: segmen ruang angkasa (*space component*), segmen kontrol (*control component*), dan segmen pengguna (*user component*), dilakukan untuk memisahkan tugas dan tanggung jawab masing-masing segmen dalam menjalankan fungsi GPS.

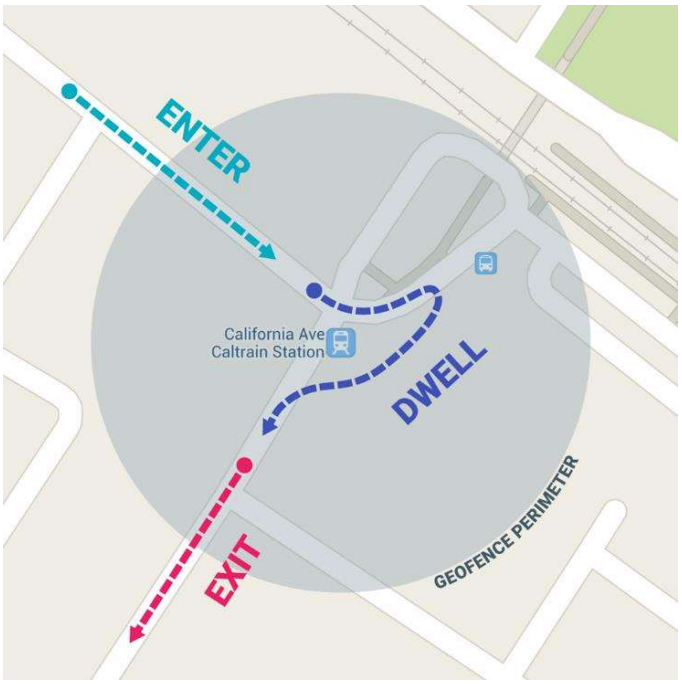
1. *Space Component*: Komponen yang terdiri dari 24 satelit GPS yang mengorbit di sekitar Bumi. Satelit mengirimkan sinyal ke penerima GPS di permukaan bumi. Sinyal yang dikirimkan berisi informasi tentang posisi satelit dan waktu sinyal dikirimkan. [16]
2. *Control Component*: Komponen ini terdiri dari beberapa stasiun kontrol di seluruh dunia yang mengontrol dan memantau satelit GPS. Stasiun kontrol ini mengirimkan sinyal ke satelit untuk melakukan koreksi waktu dan posisi satelit. Selain itu, stasiun kontrol ini juga mengumpulkan data dari satelit dan mengirimkannya ke pengguna melalui satelit. [16]
3. *User Component*: Komponen penerima GPS yang terpasang pada perangkat pengguna, seperti *smartphone*, kendaraan, atau perangkat navigasi. Penerima GPS menerima sinyal dari satelit GPS dan menghitung jarak dari satelit dengan menggunakan waktu yang dibutuhkan sinyal untuk mencapai penerima. [16]

2.8 Geofence

Geofence dapat digambarkan sebagai wilayah virtual di suatu lokasi geografis dalam area tertentu yang telah ditentukan. Terdapat dua bentuk yang dapat ditentukan dalam membatasi area, pertama dengan menggunakan jenis *Circular* yang dipetakan berbentuk lingkaran dengan pusat dan jari-jari. Adapun yang kedua, jenis *Polygon* berbentuk dari beberapa titik-titik koordinat yang ditetapkan sehingga membentuk lebih kompleks membentuk poligon. Pemilihan antara *Circular* atau *Polygon* tergantung pada kebutuhan dan karakteristik wilayah yang akan dibatasi. *Polygon* dapat digunakan untuk batas-batas yang lebih akurat, seperti dapat mengikuti bentuk tanah atau bangunan dan dapat menangani wilayah dengan bentuk yang lebih kompleks atau tidak beraturan. Sedangkan, *Circular* relatif lebih sederhana, mudah dipahami dan diimplementasikan, namun kurang akurat dalam menentukan batasan wilayah. [2], [17]



Gambar 2. 4 Contoh Visual *Circular* dan *Polygon Geofence* pada Peta Digital



Sumber gambar :
https://developer.android.com/training/location/geofencing

Gambar 2. 5 Tiga Event Utama Geofence

Pada *Geofence*, baik bentuk *Circular* maupun *Polygon* terdapat 3 kejadian utama yaitu "*Enter*", "*Dwell*", dan "*Exit*". Ketiga kejadian ini menandakan kondisi lokasi pengguna terhadap *geofence*.

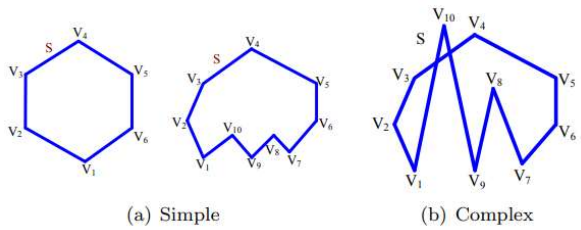
1. *Enter*: Kejadian ini terjadi ketika perangkat pengguna memasuki area *geofence* yang telah ditentukan.
2. *Dwell*: Kejadian yang terjadi ketika pengguna berada di dalam area *geofence* untuk waktu yang ditentukan. Apabila pengguna masuk maka akan terhitung berapa lama pengguna berada di dalam sebuah area *geofence*.
3. *Exit*: Event ini terjadi ketika perangkat pengguna keluar dari area geofence yang telah ditentukan.

Dalam hal ini, apabila pengguna masuk ke dalam wilayah yang telah ditetapkan, maka sistem presensi digital dapat secara otomatis membuka sebuah fitur presensi yang semula terkunci kemudian dapat melakukan presensi masuk dan presensi keluar.

2.9 Algoritma Even-Odd Rule

Menentukan apakah suatu titik berada di dalam atau di luar sebuah poligon digunakan (*Point-in-polygon*) adalah masalah geometris sederhana. Dalam banyak aplikasi ilmiah dan teknik, diperlukan untuk mengetahui apakah suatu titik tertentu terletak di dalam atau di luar poligon. Untuk poligon tertutup 'S' dan dengan sembarang titik 'P', harus ditentukan apakah titik tersebut terletak di dalam atau di luar poligon. Terutama, dalam banyak sistem berbasis grafik komputer dan sistem pemrosesan informasi geometris, perlu untuk menentukan apakah suatu titik terkandung dalam batas poligon atau tidak. Masalah ini disebut sebagai masalah inklusi titik, masalah klasik dalam grafik komputer dan geometri komputasi. E*ven-Odd Rule* adalah salah satu algoritma yang dapat sering digunakan untuk memecahkan masalah tersebut. [18], [19]

Secara umum poligon secara luas diklasifikasikan ke dalam dua kategori utama seperti yang ditunjukkan pada Gambar 2.6 Sebuah poligon disebut poligon sederhana, jika ruas garisnya tidak berpotongan, sedangkan tepi poligon kompleks berpotongan dengan tepi lainnya. Titik sudut di tuliskan dalam simbol ‘V’ sedangkan ‘S’ merupakan simbol poligon. [18]



Gambar 2. 6 Kategori Poligon

Secara deskriptif, penjelasan algoritma *Even-Odd Rule* adalah sebagai berikut :

1. Siapkan poligon ‘S’ dan titik ‘P’ yang ingin ditentukan posisinya.
2. Inisialisasi variabel hitung dengan nilai nol.
3. Lakukan iterasi terhadap semua sisi-sisi yang ada di poligon, dan lakukan pengecekan apakah garis tersebut memotong sisi poligon, jika memotong maka tambah satu ke dalam variabel hitung.

4. Setelah selesai iterasi, lakukan pengecekan:
    a. Apabila nilai hitung tersebut adalah ganjil, maka titik ‘P’ berada di dalam poligon ‘S’
    b. Apabila nilai hitung tersebut adalah genap, maka titik ‘P’ berada di luar poligon ‘S’

Kesimpulan akhir menyatakan algoritma yang diberikan dapat dengan benar menentukan banyaknya jumlah potongan dari sebuah titik yang terjadi di dalam sebuah poligon.

2.10 Flutter

Flutter pertama kali dikenalkan kepada publik pada tahun 2015 oleh Perusahaan IT terbesar didunia yaitu Google, sebuah *framework open-source* untuk mengembangkan aplikasi *multi-platform* yang terdiri dari *mobile* (Android & iOS), web, bahkan desktop dengan menggunakan satu basis bahasa pemrograman *Dart*. *Flutter* menggunakan *widget* sebagai elemen dasar dari antarmuka pengguna, memungkinkan pengembang aplikasi dapat membangun antarmuka pengguna UI yang dapat menyesuaikan kebutuhan, mudah dipelajari dan cepat. Fitur *Hot Reload* menjadi andalan kecepatan mengubah UI dalam kode secara langsung tanpa perlu memuat ulang aplikasi. Berbagai dukungan *library* maupun *packages* kian sudah berkembang dan ditampung dalam suatu tempat yang dinamakan *Pub.dev*, tentu dapat mempercepat proses pengembangan aplikasi karena cukup banyaknya layanan atau fitur-fitur yang disediakan dari pihak ketiga.[20]

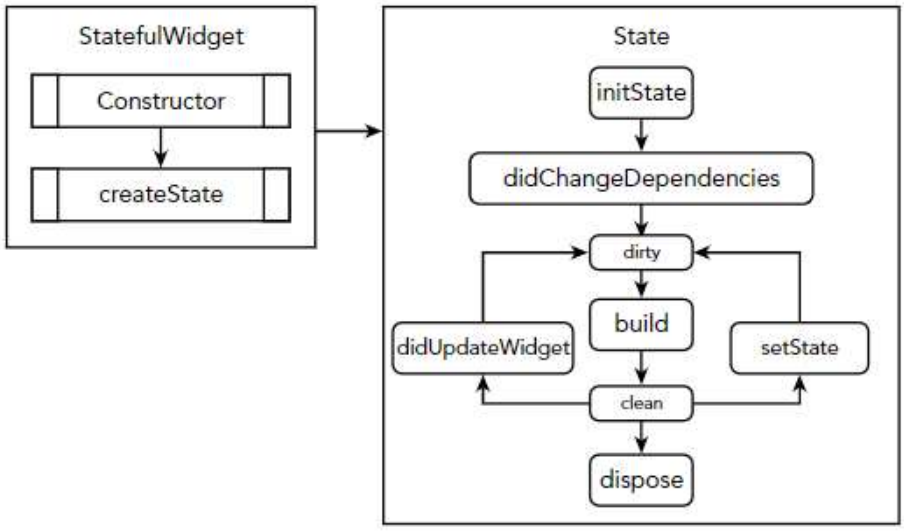
2.10.1 *Flutter Lifecycle*

Dalam bahasa pemrograman maupun sebuah kerangka kerja, masing-masing memiliki *lifecycle* yang unik dan berbeda. Untuk membangun sebuah tampilan antarmuka di kerangka kerja *Flutter,* maka kita akan mengenali istilah *Widget. Widget* dapat diartikan sebagai sebuah komponen atau bagian-bagian dari antarmuka pengguna yang dapat diatur secara terpisah dan dapat digunakan kembali pada berbagai bagian dari aplikasi. *Widget* berupa objek visual yang merepresentasikan bagian-bagian dari UI seperti tombol, teks, gambar, *textfield*, *layout* dan lain-lain. *Widget* terbagi ke dalam dua jenis, *stateless* dan *stateful*,

keduanya memiliki fungsi dan *lifecycle* yang berbeda, dapat dikatakan bahwa sebagian besar yang berada di dalam *Flutter* adalah *Widget*. [20]

1. Stateful Widget

Stateful Widget pada *Flutter* adalah jenis *Widget* yang dapat memiliki keadaan yang dapat berubah-ubah selama aplikasi dijalankan. Keadaan ini dapat berupa data atau informasi tertentu yang diambil dari pengguna atau dari sumber lainnya, dan dapat berubah seiring dengan perubahan dalam aplikasi. Ketika sebuah *Stateful Widget* diubah, *Flutter* akan secara otomatis membangun ulang *Widget* tersebut, sehingga dapat menampilkan keadaan yang baru dan membangun ulang perubahan-perubahan kecil dapat memberikan dampak yang besar pada tampilan antarmuka pengguna.[20]



Sumber gambar:
Buku *Beginning Flutter: A Hands On Guide to App Development*

Gambar 2. 7 Flutter Stateful Widget Lifecycle

Baik pada jenis *Widget Stateful* maupun *Stateless* keduanya memiliki peran penting dari *Constructor* yang merupakan spesial *method* yang akan berjalan setelah sukses menginisialisasi *class* pada saat pertama kalinya.[20]

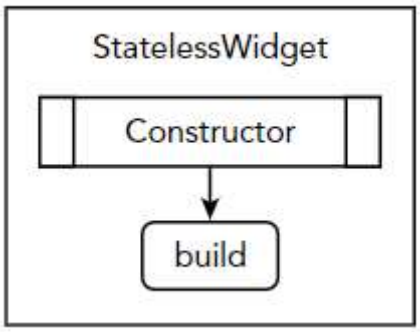
Berikut adalah deskripsi dari masing-masing metode *Lifecycle Stateful Widget* yang dijelaskan dalam bentuk tabel 2.1:

Tabel 2. 1 Daftar *method* yang tersedia pada *Stateful Widget*

Method	**Deskripsi**
createState	Dipanggil untuk menghasilkan objek *State* yang sesuai.
initState	Dipanggil hanya sekali ketika *Widget* pertama kali dibuat dan digunakan untuk inisialisasi data yang dibutuhkan oleh *Widget*.
didChangeDependencies	Dipanggil ketika keadaan dependensi dari *Widget* yang terkait berubah. Dependensi seperti variabel global, perubahan konfigurasi sistem, atau perubahan keadaan dari *Parent Widget.*
dirty	*Widget* tersebut memerlukan *update* karena adanya perubahan terkait dengan *Widget* tersebut dan perlu merekonstruksi ulang (*build*)
didUpdateWidget	Dipanggil ketika *Widget* menerima *input* baru, seperti perubahan pada properti atau *state* dan pada biasanya akan memperbaharui data atau mengubah *Widget* yang ditampilkan di layar.
build	Dipanggil setiap kali terjadi perubahan *state* pada *Widget* dan dapat dipergunakan membangun atau memperbarui *Widget* yang ditampilkan di layar.
setState	Metode ini digunakan untuk memberitahu Flutter framework bahwa keadaan (state) dari widget telah berubah, sehingga framework akan membangun ulang tampilan widget sesuai dengan keadaan terbaru.
dispose	Dipanggil ketika *Widget* dihapus dari *Widget Tree* dapat berfungsi membersihkan atau melepaskan sumber daya yang digunakan oleh *Widget.*

2. *Stateless Widget*

Lifecycle Stateless Widget pada *Flutter* merupakan siklus hidup yang sederhana dibangun berdasarkan aturan sendiri dan tidak berubah secara dinamis. Sebagai contoh, layar menampilkan sebuah gambar dengan deskripsi dan tidak akan berubah. Terdiri dari dua bagian yaitu *constructor* seperti yang sudah dijelaskan pada poin sebelumnya dan *build*() yang berguna pada saat *Widget* di inisialisasi, metode *build*() akan dipanggil untuk membuat tampilan awal dari *Widget* tersebut.[20]



Sumber gambar:
Buku Beginning Flutter: A Hands On Guide to App Development

Gambar 2. 8 *Flutter Stateless Widget Lifecycle*

2.11 React

React pertama kali diciptakan oleh Jordan Walke, seorang *software engineer* di Facebook. *React* pertama kali diterapkan pada *newsfeed* Facebook pada tahun 2011 dan kemudian pada Instagram setelah Facebook mengakuisisi Instagram pada tahun 2012. Pada acara JSConf 2013, *React* dibuat menjadi *open source*. *React* dikala itu digambarkan sebagai "V dalam MVC". Dalam arti lain, komponen *React* hanya bertindak sebagai antarmuka (UI) dan cara kerja dari komponen adalah memecah sebuah tampilan besar menjadi bagian-bagian yang lebih kecil memudahkan pengembang dalam memelihara tampilan aplikasi yang berkembang dan kompleks. Selain itu, *React* menggunakan Virtual DOM (*Document Object Model*) untuk meningkatkan performa setiap kali melakukan perubahan pada tampilan aplikasi dengan lebih cepat dan efisien.[21]

2.11.1 *React Lifecycle*

React lifecycle terdiri dari sejumlah metode yang dipanggil secara berurutan ketika suatu komponen dipasang atau diperbarui. Metode-metode ini dipanggil sebelum atau sesudah komponen *render* tampilan antarmuka. Bahkan, metode *render* sendiri merupakan bagian dari siklus hidup komponen. *React* memiliki *lifecycle* dibagi menjadi tiga fase utama: *Mounting, Updating,* dan *Unmounting*[22]. Berikut adalah penjelasan dari tiga fase yang ada pada kerangka kerja *React* untuk membangun tampilan dasar web :

1. Mounting

Lifecycle terjadi ketika sebuah *instance* komponen pertama kali dibuat dan menyisipkan elemen *html* dalam bentuk *node* ke dalam *Document Object Model* (DOM) sebagai struktur hierarki dari dokumen *html* yang memungkinkan adanya terjadinya perubahan tampilan antarmuka. Selama proses *mounting*, metode pada komponen dijalankan secara berurutan seperti metode *constructor*(), *render*(), dan *componentDidMount*(). [22]

Tabel 2. 2 *Lifecycle Mounting* di *React*

Method	**Deskripsi**
constructor	Dipanggil untuk menginisialisasi keadaan *(state)* dan melakukan *bind* pada *event handler* yang berguna untuk menghindari kesalahan pada akses *state* atau *properti* dari komponen yang salah.
render	Menghasilkan keluaran (*output*) dalam bentuk virtual DOM. Virtual DOM merupakan konsep *React* yang membandingkan perbedaan dan memperbaharui bagian yang berubah tanpa memuat ulang halaman.
React updates Dom and Refs	*React* akan membuat perubahan pada DOM dan memperbaharui nilai *refs* sesuai dengan representasi virtual terbaru. *Refs* adalah cara mengakses *instance* komponen pada *React.*

componentDidMount	Dipanggil ketika komponen sudah siap ditampilkan di halaman web.

2. Updating

Lifecycle ini dapat diartikan sebagai proses yang terjadi ketika komponen menerima perubahan pada *props* atau *state*. *Props* berguna sebagai parameter untuk mengirimkan data dari suatu komponen ke komponen lain. *State* adalah sebuah objek yang menggambarkan bagaimana komponen harus di *render*. [22]

Tabel 2. 3 *Lifecycle Updating* di *React*

Method	**Deskripsi**
render	Proses ini menyerupai saat *mounting*. namun akan berjalan apabila terjadi perubahan *props* baru yang diberikan ke dalam suatu komponen, mengubah nilai *state* dan terjadi *forceUpdate* atau metode yang memaksa komponen harus di *render* ulang.
React updates Dom and Refs	Proses ini sama seperti yang dilakukan ketika *mounting*. Di mana *React* memperbaharui DOM dan *Refs*.
componentDidUpdate	*Lifecycle* yang digunakan untuk menangani apabila terjadi perubahan pada *props* atau *state* dari komponen.

3. Unmounting

Unmounting adalah fase terakhir dalam siklus hidup komponen *React*. *Lifecycle* yang terjadi ketika komponen dihapus dari DOM. Ketika fase ini terjadi, metode *componentWillUnmount*() dipanggil, dan hanya ada satu *method* saja.[22]

Tabel 2. 4 *Lifecycle Unmounting* di *React*

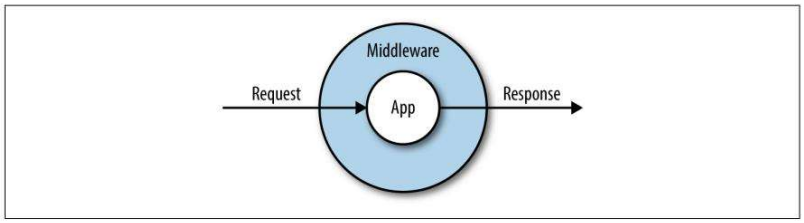
Method	**Deskripsi**
componentWillUnmount	React akan membersihkan sumber daya atau *event listener* yang telah digunakan oleh komponen sebelum dihapus untuk mengoptimalkan kinerja aplikasi dan memastikan tidak terjadi pemborosan sumber daya yang tidak diperlukan.

2.12 Laravel

Laravel merupakan sebuah *framework* PHP yang *powerful* sekaligus fleksibel, karena memiliki komunitas terus berkembang dan ekosistem yang cukup luas. *Laravel* dikembangkan oleh Taylor Otwell, dirilis pada tahun 2011 dan saat ini *Laravel* telah menyentuh versi 10. *Framework* ini menggunakan pendekatan *Model-View-Controller* (MVC) sebagai inti dari pengembangan aplikasi. Tak hanya itu *Laravel* menyediakan berbagai ekosistem yang mempermudah pengembangan aplikasi, seperti fitur *routing, middleware, database migration, authentication, authorization*, dan *testing*. Pada intinya tujuannya dari *Laravel* dibangun adalah menyediakan kode yang jelas, sederhana, dan rapih beserta fitur-fiturnya yang membantu para pengembang untuk belajar, memulai, mengembangkan, dan menulis kode yang sederhana, jelas, dan tahan lama.[23]

2.12.1 *Laravel Request Lifecycle*

Lifecycle ini dijelaskan untuk memperkirakan dan memahami alur data dan logika berjalan, *Laravel* memiliki tiga siklus yang terdiri: *Request, Middleware*, dan *Response*. *Laravel Request Lifecycle* ini bersifat unik, terbatas dan hanya akan berjalan apabila kerangka *Laravel* dipergunakan sebagai *web service* bukan sebagai kerangka kerja *Full Stack Web.* [23]



Sumber Gambar:

Buku Laravel: Up & Running, 2nd Edition

Gambar 2. 9 Laravel Request Lifecycle

Tahap pertama yaitu *Request*, setiap kali *Laravel* menerima permintaan HTTP dari *client*, kemudian *router* di dalam *Laravel* akan mengalamatkan atau menentukan *controller* dan *method* yang akan menangani permintaan tersebut. Kemudian, permintaan akan melewati serangkaian *middleware* yang dapat melakukan berbagai tindakan, seperti verifikasi pengguna, manipulasi permintaan, atau memodifikasi *response* yang dihasilkan. Terakhir, ketika permintaan selesai diproses oleh *middleware*, *Laravel* akan menghasilkan *response* HTTP yang dikirimkan kembali ke *client* dan biasanya membutuhkan pemrosesan yang berkaitan dengan *database*. [23]

2.13 Javascript Object Notation

JSON (*JavaScript Object Notation*) merupakan format pertukaran data antara satu sistem dengan sistem lainnya, JSON memiliki ukuran yang sangat ringan (*lightweight*) mudah dibaca oleh para pengembang perangkat lunak. Format JSON merupakan cara merepresentasikan dan menyimpan data dengan menggunakan struktur objek atau *array* yang di dalamnya terdiri dari pasangan *key-value*. JSON memiliki kemudahan dalam mempertukarkan informasi antara aplikasi atau platform yang berbeda secara efisien, contohnya platform Android melakukan permintaan data (*request*) kepada Web Service, kemudian Web Service akan mengirimkan *response* sesuai kebutuhan permintaan. [24]

2.14 *Unified Modeling Language* (UML)

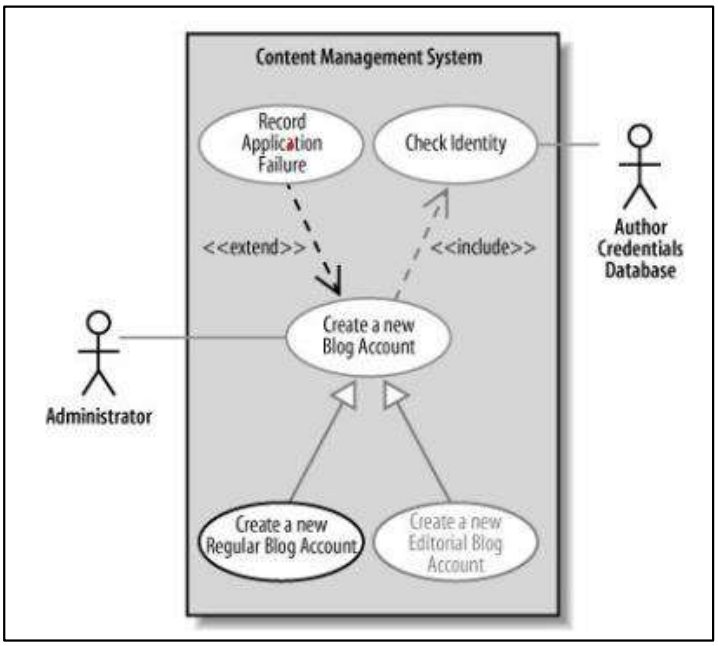
Unified Modeling Language (UML) adalah sebuah bahasa pemodelan visual yang digunakan untuk menggambarkan dan merancang sistem perangkat lunak.

UML digunakan untuk menggambarkan, merancang, dan memodelkan sistem perangkat lunak secara terstruktur dan sistematis. UML bermanfaat untuk melihat gambaran besar dari suatu masalah, sehingga dapat mengurangi kompleksitas sistem dengan membaginya menjadi bagian-bagian yang lebih kecil, sehingga memudahkan dan analisis dan memberikan cara yang konsisten untuk dokumentasi tentang sistem sehingga dapat mempermudah anggota tim dalam membangun proyek perangkat lunak untuk memberikan gambaran cara kerja, memahami dan dapat bekerja dengan model yang sama. [25]

2.14.1 *Use Case Diagram*

Sebuah diagram didesain untuk memberikan bagaimana interaksi antara pengguna dan sistem yang terjadi di dalam suatu sistem perangkat lunak. *Use case diagram* terdiri dari *use case* yang merupakan fungsi yang dapat dilakukan oleh sistem, aktor sebagai pengguna atau sistem eksternal yang terlibat dalam interaksi dengan sistem, dan relasi antara keduanya. Terdapat tiga relasi yang digunakan antara *use case* dan aktor seperti *include* adalah menggabungkan fungsionalitas sehingga menjadi bagian dari *use case* lain, *extend* menambahkan fungsionalitas opsional dalam kondisi tertentu dan *generalization* menunjukkan hubungan antara *use case* yang memiliki karakteristik dan fungsionalitas yang sama [25].

Berikut adalah gambaran *use case diagram*, terdapat sebuah aktor yaitu *user* yang dapat melakukan berbagai *use case*:



Sumber Gambar:

Buku *Learning* UML 2.0

Gambar 2. 10 Use Case Diagram

2.14.2 *Activity Diagram*

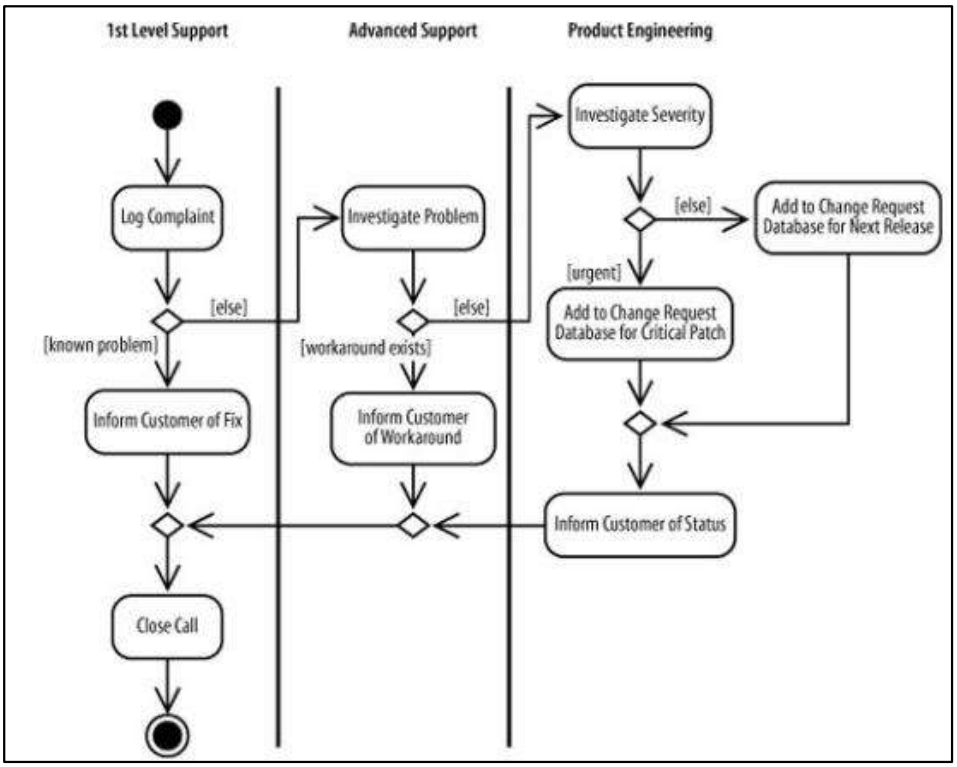
Activity Diagram digunakan dalam pemodelan proses bisnis atau pemodelan sistem perangkat lunak untuk menggambarkan urutan aktivitas tindakan-tindakan tingkat tinggi (*high level*) yang dihubungkan bersama menjadi satu kesatuan untuk mewakili proses yang terjadi dalam sistem. Komponen-komponen penting yang ada pada *activity diagram* :

1. *Start node,* untuk mengindikasikan permulaan aktivitas.
2. *Action*, sebagai sebuah aksi yang berjalan.
3. *Decision,* dipergunakan untuk membuat sebuah keputusan berdasarkan kondisi.
4. *Merge,* berfungsi untuk menggabungkan jalur menjadi aliran tunggal.
5. *Fork,* membagi satu jalur aliran menjadi beberapa jalur aliran yang saling independen.
6. *Join,* menggabungkan beberapa jalur aliran yang sebelumnya telah dibagi dengan *fork* menjadi satu jalur aliran tunggal.

7. *Swimlanes,* membagi diagram menjadi beberapa kolom dan masing-masing diagram memiliki aktivitas atau tanggung jawab yang berbeda.
8. *Activity final node,* indikator dari akhir aktivitas.

Masing-masing komponen memiliki fungsi dan simbol yang berbeda untuk menggambarkan alur kerja dari sistem. [25]

Berikut dilampirkan gambar aktivitas diagram yang membagi tanggung jawab masing-masing dari pembeli maupun sistem, terlihat juga adanya kondisi di mana memengaruhi alur dari tindakan selanjutnya.



Sumber Gambar:

Buku Learning UML 2.0

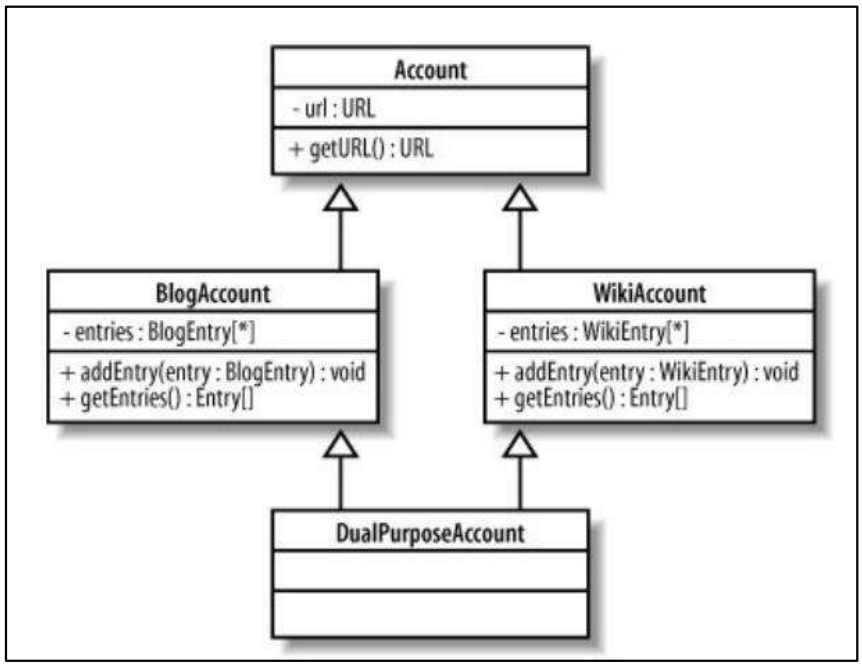
Gambar 2. 11 *Activity Diagram*

2.14.3 *Class Diagram*

Sebuah diagram yang merepresentasikan struktur dari sebuah sistem atau aplikasi yang terdiri dari beberapa kelas-kelas dan memiliki hubungan antara masing-masing kelas tersebut. Komponen *class* merupakan elemen dasar digambarkan sebagai objek atau entitas yang dapat memiliki atribut sebagai

properti yang dimiliki dan *method* sebagai tindakan apa yang bisa dilakukan oleh *class*. Masing-masing properti dan *method* memiliki konsep *Visibility* yang memberikan aturan siapa yang dapat mengakses properti dan metode. serta dapat dihubungkan dengan kelas-kelas lain melalui hubungan seperti *Association*, *Aggregation*, dan *Composition*. *Association* menggambarkan bahwa kelas dapat berhubungan dengan kelas lain dan memiliki nama dan arah untuk memperjelas hubungan antara kelas, *Aggregation* menggambarkan bahwa sebuah *class* terdiri dari beberapa entitas sedangkan *Composition* menyatakan bahwa sebuah *class* tidak dapat berdiri sendiri tanpa adanya entitas yang menjadi bagiannya. Adapun *class* dapat melakukan *Inheritance* untuk menyatakan bahwa sebuah *class* dapat memberikan pewarisan properti atau *method* kepada *class* yang berada di bawahnya. [25]

Berikut gambar 2.12 adalah hasil *Class Diagram* untuk merepresentasikan kelas-kelas juga memberikan relasi antar kelas.



Sumber Gambar:

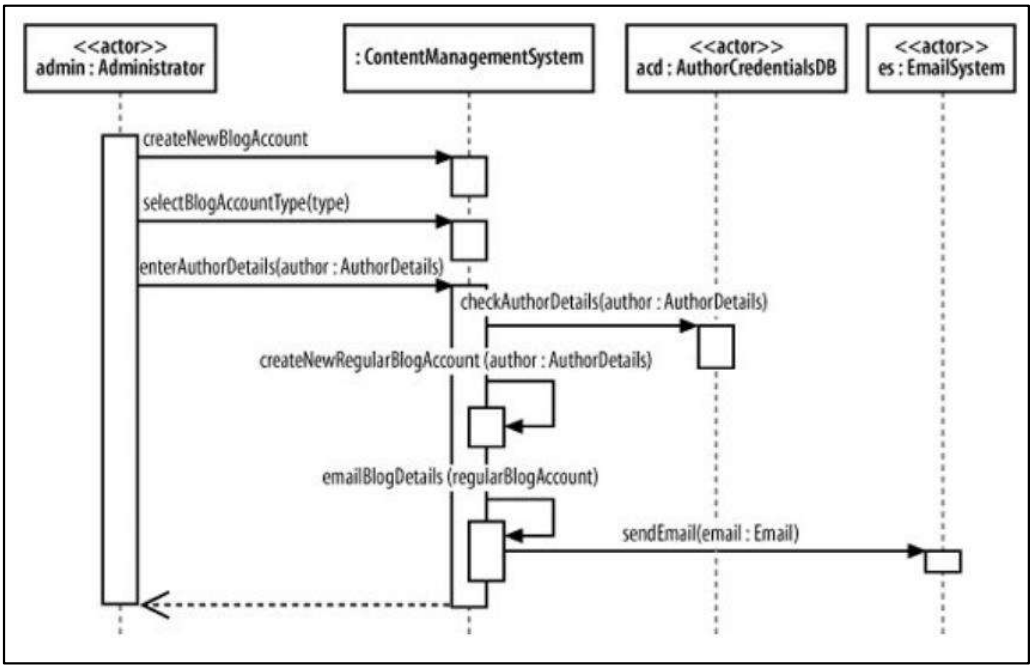
Buku Learning UML 2.0

Gambar 2. 12 *Class Diagram*

2.14.4 *Sequence Diagram*

Sequence Diagram digunakan untuk memodelkan apa yang harus dilakukan oleh sistem, sebuah interaksi antara objek dalam *use case* tertentu. *Sequence diagram* juga menjelaskan interaksi apa yang akan dipicu ketika suatu kasus penggunaan *use case* tertentu dieksekusi dan dalam urutan apa interaksi tersebut akan terjadi. Komponen-komponen penting dalam *sequence diagram* meliputi aktor atau objek digambarkan dalam persegi panjang dengan nama sebagai entitas yang berpartisipasi dalam interaksi sistem, *lifeline* menunjukkan waktu di mana objek aktif berinteraksi dalam sistem, anak panah merepresentasikan oleh pesan yang akan dikirim ke objek lain, *Return Message* berguna untuk pesan yang dikirim sebagai respons dari pesan sebelumnya, *Self Message* berfungi pesan yang dikirim ke aktor atau objek sendiri. [25]

Sequence diagram 2.13 tentang memperlihatkan proses yang terjadi di dalam sistem dan setiap objek berinteraksi satu sama lain sebagai satu bagian urutan.



Sumber Gambar:

Buku *Learning* UML 2.0

Gambar 2. 13 *Sequence Diagram*

2.15 Metode Likert

Skala Likert merupakan salah satu metode alat ukur yang diandalkan untuk mengukur tingkat kepuasan dan pandangan pengguna terhadap sebuah aplikasi. Skala Likert digunakan untuk mengukur sikap, pendapat dan persepsi seseorang tentang variabel penelitian. Penggunaan skala Likert variabel yang ukur dijadikan sebuah indikator untuk titik tolak guna menyusun pernyataan atau pertanyaan, dalam hal ini kenyataan bahwa pengujian aplikasi dapat diterima atau sesuai dengan kebutuhan pengguna aplikasi[26]. Jawaban yang disusun memiliki gradasi dari item sangat positif atau berupa kata-kata lain di antara-nya:

a. Sangat setuju, setuju, rata-rata, tidak setuju, sangat tidak setuju

b. Selalu, sering, kadang-kadang, tidak pernah

c. Sangat positif, positif, negative, sangat negative

d. Sangat baik, baik, tidak baik, sangat tidak baik