BAB 2 TINJAUAN PUSTAKA

2.1. Tempat Penelitian

Proses penelitian ini dilakukan di sebuah tempat penelitian yang memiliki permasalahan untuk mendukung jalannya proses penelitian. Tempat penelitian tersebut adalah UD Sinar Jaya Padang. UD Sinar Jaya Padang merupakan sebuah perusahaan distributor yang berdiri sejak tahun 2016. UD Sinar Jaya Padang melakukan pendistribusian produk dari beberapa manufaktur dengan beragam jenis produk. UD Sinar Jaya Padang bertugas menjadi penghubung antara manufaktur atau pabrik dengan toko atau retail.

Berikut ini adalah perincian alamat dan kontak tempat penelitian.

Lokasi Kantor:

Jalan Pasar Malintang Dalam No. 3, Padang, Sumatera Barat, Indonesia

Kode Pos Kantor:

25211

Kontak:

Instagram: @sinar_jaya_pdg

Email: sinarjayapdgud@gmail.com

2.1.1. Logo Tempat Penelitian

Logo merupakan sebuah gambar atau sketsa yang memiliki arti tertentu untuk mewakili citra sebuah organisasi atau instansi. Untuk menggambarkan citranya di publik, maka UD Sinar Jaya Padang juga memiliki sebuah logo. Logo tersebut bisa dilihat pada gambar 2.

Gambar 2. Logo UD Sinar Jaya Padang

Berdasarkan gambar 2 di atas, maka logo tersebut memiliki arti sebagai berikut.

1) Warna hijau melambangkan sumber kehidupan, kesegaran, dan rasa aman. Warna hijau ini juga digunakan oleh perusahaan dengan harapan perusahaan dapat tumbuh subur seperti tanaman.
2) Warna merah melambangkan kehangatan, semangat, gairah, dan energi. Perusahaan menggunakan warna merah dengan harapan dapat memberikan kehangatan bagi karyawan. Selain kehangatan, perusahaan juga menggunakan warna merah dengan harapan perusahaan akan terus bergairah dan bersemangat saat beroperasi.
3) Perusahaan menggambarkan logonya berdasarkan singkatan nama perusahaan, yaitu "SJ". Huruf S diberi warna merah dan berada di atas huruf J. Huruf J diletakkan dengan bentuk ujung yang bulat menyerupai wadah, yang artinya bahwa seluruh sinar (melambangkan rezeki) dapat ditampung oleh perusahaan.
4) Pada logo juga terdapat nama perusahaan "SINAR JAYA".

2.1.2. Visi dan Misi Tempat Penelitian

Visi dan Misi merupakan sebuah hal yang penting dalam jalannya sebuah perusahaan. Seluruh kegiatan yang dilakukan oleh perusahaan akan berpegang pada visi dan misi perusahaan.

Adapun visi yang dimiliki oleh perusahaan distributor UD Sinar Jaya Padang adalah "Meningkatkan pendistribusian barang yang terkemuka dan terpercaya untuk mencapai profit atau laba yang besar".

Sedangkan untuk misi yang dimiliki oleh perusahaan distributor UD Sinar Jaya Padang adalah sebagai berikut:

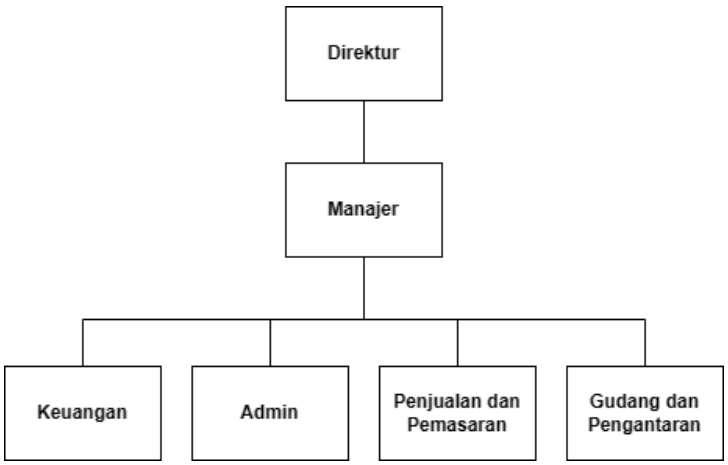
1) Memberikan produk berkualitas dengan harga kompetitif dan bermanfaat.

2) Memberikan pelayanan yang optimal dan profesional untuk meningkatkan kepercayaan dan menjaga hubungan yang baik dengan mitra kerja.
3) Membimbing sumber daya manusia dengan pengembangan diri.
4) Menjaga, mengawasi, memastikan produk sampai dalam kondisi baik saat distribusi.
5) Menggunakan sistem teknologi yang tepat.

2.1.3. Struktur Organisasi Tempat Penelitian

Struktur organisasi adalah sebuah sistem hierarki atau jenjang yang digunakan dengan tujuan agar sebuah perusahaan akan lebih terorganisir dengan pembagian kerja, tugas, dan tanggung jawab yang berbeda-beda. Struktur organisasi diharapkan akan mampu membuat sumber daya manusia yang ada di perusahaan dapat saling bekerja sama.

Adapun struktur organisasi pada UD Sinar Jaya Padang adalah seperti gambar 3 berikut ini.



Gambar 3. Struktur Organisasi UD Sinar Jaya Padang

Pada perusahaan UD Sinar Jaya Padang terdapat 4 (empat) divisi, yaitu:

A. Divisi Keuangan

Divisi keuangan adalah divisi yang bertugas untuk mengelola dan menghitung masuk dan keluarnya dana perusahaan. Divisi ini akan berisi

Akuntan, Perencana dan Pengelola Anggaran. Divisi ini harus memastikan kegiatan berikut terlaksana dengan baik, seperti:

1. Menyusun dan melakukan verifikasi laporan keuangan
2. Melakukan pembayaran tagihan dan kebutuhan perusahaan
3. Membuat analisis keuangan
4. Membuat laporan jurnal
5. Manajemen keuangan perusahaan

B. Divisi Administrasi

Divisi administrasi adalah divisi yang bertugas untuk membantu eksekutif dalam hal mengorganisir dan memantau tugas administratif. Divisi ini akan berisi Admin. Divisi ini harus memastikan kegiatan berikut terlaksana dengan baik, seperti:

1. Mengelola dokumen perusahaan
2. Melakukan *entry* data
3. Mengarsipkan dokumen
4. Melakukan koordinasi dengan divisi lain

C. Divisi Penjualan dan Pemasaran

Divisi penjualan dan pemasaran adalah divisi yang bertugas untuk meningkatkan penjualan dan memasarkan produk atau layanan perusahaan. Divisi ini akan berisi *Supervisor* dan *Sales*. Divisi ini harus memastikan kegiatan berikut terlaksana dengan baik, seperti:

1. Melakukan riset pasar untuk memahami kebutuhan dan preferensi konsumen
2. Mengembangkan strategi penjualan
3. Mengembangkan strategi pemasaran
4. Mengembangkan target pasar
5. Mengumpulkan dan menganalisis data penjualan untuk mengidentifikasi tren penjualan

D. Divisi Gudang dan Pengantaran

Divisi gudang dan pengantaran (logistik) adalah divisi yang bertugas untuk mengelola rantai pasokan dan memastikan pengiriman produk atau

layanan kepada pelanggan dengan efisien. Divisi ini akan berisi Kepala Gudang, Analis Logistik, *Quality Assurance & Control*, dan Pengantaran. Divisi ini harus memastikan kegiatan berikut terlaksana dengan baik, seperti:

1. Mengelola stok persediaan perusahaan
2. Memproses pesanan pelanggan dan mempersiapkan pengiriman produk atau layanan
3. Mengatur pengiriman barang kepada pelanggan
4. Mengatur pengiriman barang kepada pelanggan
5. Memastikan kualitas barang yang dikirim kepada pelanggan
6. Mengidentifikasi dan mengelola risiko yang terkait dengan logistik dan pengantaran

2.2. Landasan Teori

Landasan teori adalah teori-teori yang digunakan pada penelitian yang diyakini mampu menjadi referensi untuk menunjang dan memperdalam pemahaman. Adapun landasan teori yang digunakan adalah sebagai berikut.

2.2.1. Aplikasi

Berdasarkan KBBI, kata aplikasi sering digunakan untuk menggambarkan sebuah penerapan atau penggunaan. Namun dalam ilmu teknologi, aplikasi memiliki arti yang berbeda. Aplikasi adalah sebuah program yang dibuat untuk kebutuhan tertentu berdasarkan pemakainya [5]. Sedangkan pengertian aplikasi menurut *Buyens* (2001) adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas. Aplikasi memiliki tujuan untuk mempermudah kegiatan dari pemakai. Aplikasi perangkat lunak secara umum terbagi atas 3 tipe platform [6], yaitu:

1. *Mobile*

   Aplikasi *mobile* adalah aplikasi yang biasanya dijalankan pada *mobile* atau *smartphone*. Adapun *mobile* ini biasanya menggunakan sistem operasi seperti, *Android* dan *iOS*.

2. *Desktop*

   Aplikasi *desktop* adalah sebuah aplikasi yang biasanya dijalankan pada program komputer atau laptop. Aplikasi *desktop* ini bisa digunakan jika pada perangkat komputer atau laptop tersebut telah terpasang aplikasinya.

3. *Website*

   Aplikasi *website* merupakan aplikasi yang paling fleksibel. Aplikasi ini bisa dibuka dari perangkat mana saja dengan syarat bahwa perangkat tersebut bisa mengakses internet dan memiliki *browser* di dalamnya. Aplikasi *website* ini akan diakses menggunakan alamat *url*.

2.2.2. Aktivitas

Aktivitas sering juga disebut sebagai kegiatan. Pada KBBI, aktivitas diartikan sebagai kerja atau salah satu kegiatan kerja yang dilaksanakan dalam tiap bagian di dalam perusahaan. Sedangkan menurut Anton Mulyono (2001) aktivitas artinya kegiatan atau keaktifan jadi segala sesuatu yang dilakukan atau kegiatan-kegiatan yang terjadi baik fisik maupun non-fisik merupakan suatu aktivitas.

2.2.3. Sales

Menurut ilmu *marketing, Sales* adalah kegiatan pada perusahaan yang bertujuan untuk meningkatkan penjualan melalui pemasaran. *Sales* biasanya dilakukan dengan memanfaatkan promosi [7].

2.2.4. Aktivitas *Sales*

Aktivitas *sales* adalah sebuah aktivitas yang dilakukan oleh pelaku pemasaran dan penjualan, yang biasanya sering disebut sebagai *salesman* atau *salesperson*. *Salesman* akan melakukan aktivitas promosi penjualan untuk membuat konsumen tertarik membeli barang yang ditawarkan oleh perusahaan. Aktivitas *sales* ini diharapkan mampu menambah profit sebuah perusahaan [8].

Salesman memiliki tugas dan tanggung jawab yang harus dijalaninya selama melakukan aktivitas *sales*. Adapun tugas dan tanggung jawab tersebut adalah sebagai berikut:

1. Menjual produk

   Tugas ini merupakan tugas paling mendasar dari seorang *salesman*. *Salesman* harus menjual produk berdasarkan atas target yang telah diberikan.
2. Kebutuhan administratif pelanggan

   Tidak hanya menjual produk, *salesman* juga harus memastikan kebutuhan pelanggan secara administratif terpenuhi. Kebutuhan administratif misalnya berupa bukti pembelian pelanggan.
3. Membuka peluang pasar yang baru

   Salesman bukan hanya menawarkan produk kepada pelanggan-pelanggan yang sudah berlangganan, tetapi *salesman* juga harus membuka peluang pasar dengan cara menjangkau pasar-pasar yang belum pernah membeli produk.
4. Mengikuti dinamika pasar

   Saat menawarkan produk, *salesman* harus bisa memperkirakan dan menyesuaikan pasar yang ditawarkan dengan tren yang sesuai dengan lokasi pasar. *Salesman* harus bisa mengenal tren pasar yang sedang berlangsung sebelum menawarkan produk pada pasar.
5. Layanan purna jual

   Tugas *salesman* bukan hanya mengenai penjualan, tetapi juga menjaga hubungan yang baik dengan pelanggan. Layanan purna jual ini bisa berupa menerima pertanyaan pelanggan, menerima keluhan pelanggan, menerima masukan dan kritikan pelanggan, hingga pengembalian barang ketika produk dinilai tidak layak untuk dijual berdasarkan standar yang telah ditetapkan perusahaan.
6. Menjaga reputasi perusahaan

Salesman adalah orang yang paling sering berhubungan langsung dengan pelanggan. Oleh karena itu untuk menjaga reputasi perusahaan, *salesman* harus berperilaku baik untuk menjaga nama baik perusahaan.

2.2.5. Aplikasi Aktivitas *Sales*

Aplikasi aktivitas *sales* adalah sebuah aplikasi yang digunakan dalam aktivitas *sales* sebuah perusahaan. *Salesman* akan melakukan aktivitasnya dalam kegiatan *sales* dengan pemanfaatan aplikasi. Aplikasi aktivitas *sales* ini akan mempermudah kegiatan *salesman* dalam melaksanakan tugas dan tanggung jawabnya. Aplikasi ini akan membantu dalam beberapa proses yang menjadi kesulitan perusahaan dan *salesman*[9].

2.2.6. Android

Android adalah sebuah sistem operasi *mobile* berdasarkan versi modifikasi *Linux* yang dirancang terutama untuk perangkat seluler layar sentuh, seperti *smartphone*, *tablet*, dan *smartTV* [10]. *Android* menyediakan platform *open source* yang bisa digunakan developer untuk mengembangkan aplikasi berbasis *android*. *Android* awalnya didirikan pada tahun 2003, namun dibeli oleh *Google* pada tahun 2005.

2.2.6.1. Arsitektur *Android*

Sistem operasi *Android* merupakan kumpulan komponen perangkat lunak. Komponen utama Arsitektur Sistem Operasi *Android* atau *Software Stack* adalah *Linux kernel*, *native libraries*, *Android Runtime*, *Application Framework* dan *Applications* [11].

1. *Linux Kernel*

   Linux Kernel berada di lapisan paling bawah dari *software stack*. Seluruh Sistem Operasi *Android* dibangun di atas lapisan ini dengan beberapa perubahan yang dilakukan oleh pihak *Google*. Seperti sistem operasi utama, mereka menyediakan fungsionalitas berikut: manajemen proses, manajemen memori, manajemen perangkat (misalnya kamera, *keypad*, tampilan, dll.). Sistem operasi *Android* berinteraksi dengan

perangkat keras dari *device* dengan lapisan ini. Lapisan ini juga berisi banyak *driver* perangkat keras penting. *Linux Kernel* juga bertanggung jawab untuk mengelola memori virtual, jaringan, *driver*, dan manajemen daya.

2. *Native Libraries Layer*

   Di bagian atas lapisan *Linux Kernel* ada *Native Library Android*. Lapisan ini memungkinkan perangkat untuk menangani berbagai jenis data, seperti data khusus untuk perangkat keras. Semua *library* ini ditulis dalam bahasa c atau c++. *Library* ini akan dipanggil melalui antarmuka *java*. Beberapa *native library* yang penting adalah:

   a) *Surface Manager*

      Digunakan untuk mengatur tampilan perangkat. *Surface Manager* berfungsi untuk menyusun tampilan jendela di layar.

   b) *SQLite*

      SQLite adalah basis data yang digunakan di *android* untuk penyimpanan data. Ini adalah basis data relasional dan tersedia untuk semua aplikasi.

   c) *WebKit*

      Ini adalah *browser engine* yang digunakan untuk menampilkan konten *HTML*.

   d) Media *framework*

      Media *framework* menyediakan pemutaran dan perekaman berbagai format audio, video dan gambar. Misalnya *MP3, AAC, AMR, JPG, MPEG4, H.264*, dan *PNG*.

   e) *Free Type*

      Untuk *Bitmap* dan *Front Rendering*.

   f) *OpenGL | ES*

      Digunakan untuk me*render* konten grafik *2D* atau *3D* ke layar.

   g) *Libc*

      Ini berisi *library* C terkait sistem.

3. *Android Runtime*

Android Runtime terdiri dari *Dalvik Virtual Machine* dan *Core Java library*. Itu terletak di tingkat yang sama dengan lapisan *library*. *Dalvik Virtual Machine* dikembangkan oleh *Dan Bornstein* dari *Google*. *Dalvik Virtual Machine* adalah salah satu jenis *Java Virtual Machine* yang digunakan untuk menjalankan aplikasi di perangkat *Android*. *Dalvik Virtual Machine* memungkinkan setiap aplikasi *Android* untuk berjalan dalam prosesnya sendiri, dengan *instance Dalvik Machine Virtual*-nya sendiri. *Dalvik Virtual Machine* memungkinkan beberapa *instance* dari *Virtual Machine* dibuat secara bersamaan dengan memberikan keamanan, isolasi, manajemen memori, dan dukungan *threading*. Tidak seperti *Java Virtual Machine* yang berbasis proses, *Dalvik Virtual Machine* ini berbasis register. *Dalvik Virtual Machine* menjalankan *file .dex* yang dibuat dari *file .class* dengan *dx tool*. *Dx tool* termasuk dalam *Android SDK*. *Dalvik Virtual Machine* dioptimalkan untuk daya pemrosesan yang rendah dan lingkungan dengan memori yang rendah.

4. *Application Framework*

   Lapisan *Application Framework* menyediakan banyak layanan tingkat tinggi atau *API* utama untuk aplikasi dalam bentuk kelas *Java*. *Developer* diizinkan untuk menggunakan layanan ini dalam aplikasi mereka. Lapisan ini merupakan blok yang berinteraksi langsung dengan aplikasi *developer*. Blok-blok penting dari *Application Framework* adalah:

   a) *Activity Manager*

      Blok ini akan mengelola siklus hidup dari aplikasi.

   b) *Content Providers*

      Blok ini digunakan untuk mengelola pembagian data antar aplikasi, mengelola cara mengakses data dari aplikasi lain.

   c) *Telephony Manager*

      Blok ini mengelola semua fungsi terkait panggilan suara.

   d) *Location Manager*

Blok ini digunakan untuk manajemen lokasi, menggunakan *GPS* atau *cell tower*.

e) *Resource Manager*

Blok ini digunakan untuk mengelola berbagai jenis sumber daya yang digunakan dalam aplikasi.

5. *Application Layer*

Applications Layer adalah lapisan paling atas dalam arsitektur *Android*. Beberapa aplikasi sudah di-*instal* sebelumnya dengan setiap perangkat, seperti: aplikasi *SMS client*, *Dialer*, *Web browser*, dan *Contact manager*. Seorang *developer* dapat menulis aplikasinya sendiri dan dapat menggantinya dengan aplikasi yang sudah ada.

2.2.7. Web

Web atau *World Wide Web* adalah sebuah halaman dengan informasi tertentu yang saling terhubung dan bisa diakses melalui internet. Dengan *web*, pengguna bisa melakukan akses, pencarian, dan interaksi dengan berbagai jenis informasi, konten dan layanan lainnya.

Dasar dari web adalah *hypertext*. *Hypertext* ini akan membuat halaman-halaman *web* saling terhubung melalui tautan (*link*) yang menghubungkan beberapa halaman.

2.2.8. GPS

Global Positioning System atau *GPS* adalah sistem satelit navigasi global yang menyediakan sinkronisasi lokasi, kecepatan, dan waktu. *GPS* adalah salah satu teknologi yang banyak digunakan pada keseharian manusia, misalnya dalam hal *tracking* transportasi [12], berbagi lokasi, dan sebagainya. *GPS* adalah sebuah teknologi dan infrastruktur berbasis satelit yang dirancang untuk secara akurat menentukan dan mengkomunikasikan posisi geografis yang tepat dari objek atau perangkat di permukaan bumi [13].

GPS, yang awalnya dikenal sebagai Sistem Pemosisian Global *Navstar*, dimulai sebagai program teknis bersama sipil/militer pada tahun 1973. Program bersama tersebut menggabungkan aspek-aspek terbaik dari beberapa kemampuan

yang berpusat pada layanan termasuk *TRANSIT*, *TIMATION*, dan *Project 621B* untuk mengurangi proliferasi alat bantu navigasi.

GPS merupakan sistem navigasi radio berbasis ruang angkasa yang terdiri dari konstelasi sinyal navigasi penyiaran satelit & jaringan stasiun bumi dan stasiun kontrol satelit yang digunakan untuk pemantauan dan kontrol. Saat ini 31 satelit *GPS* mengorbit Bumi pada ketinggian sekitar 11.000 mil yang memberikan pengguna informasi akurat tentang posisi, kecepatan, dan waktu di mana pun di dunia dan dalam segala kondisi cuaca.

2.2.8.1. Tiga Segmen *GPS*

Sistem *GPS* memiliki 3 segmen penting, yaitu satelit, pengontrol, dan pengguna (penerima) [14].

1. Segmen Satelit

   Satelit mengelilingi bumi, mentransmisikan sinyal ke pengguna pada posisi geografis dan waktu. Terdiri dari minimal 24 satelit operasional dalam enam orbit melingkar 20.200 km (10.900 NM) di atas bumi dengan sudut inklinasi 55 derajat dengan periode 11 jam 58 menit. Meskipun ini bukan persyaratan yang dinyatakan, biasanya satelit ditempatkan di slot orbit utama sehingga setiap saat minimal 6 satelit dapat dilihat oleh pengguna di mana pun di dunia.
2. Segmen Pengontrol

   Segmen Pengontrol terdiri dari stasiun monitor di Bumi, stasiun kontrol utama, dan antena di bumi. Kegiatan kontrol itu termasuk melacak dan mengoperasikan satelit di ruang angkasa dan memantau transmisi. Stasiun monitor melacak semua satelit *GPS* yang terlihat dan mengumpulkan informasi mulai dari siaran satelit. Stasiun monitor mengirimkan informasi yang mereka kumpulkan dari masing-masing satelit kembali ke stasiun kontrol utama, yang menghitung orbit satelit dengan sangat tepat. Informasi tersebut kemudian diformat menjadi pesan navigasi yang diperbarui untuk setiap satelit. Informasi yang diperbarui ditransmisikan ke setiap satelit melalui antena darat, yang juga mengirim dan menerima

kontrol satelit dan sinyal pemantauan. Ada stasiun pengontrol yang berada hampir setiap benua di dunia, termasuk Amerika Utara dan Selatan, Afrika, Eropa, Asia, dan Australia.

3. Segmen Pengguna (Penerima)

   Segmen Pengguna terdiri dari penerima, prosesor, dan antena yang memungkinkan operator darat, laut, atau udara untuk menerima siaran satelit *GPS* dan menghitung posisi, kecepatan, dan waktu yang tepat. Biasanya yang berperan penting sebagai penerima adalah item-item seperti *smartwatch*, *smartphone*, dan perangkat telematika.

2.2.9. Google Maps API

Google Maps API adalah sebuah *API* yang disediakan oleh *Google* untuk mengintegrasikan fitur-fitur data dari *Google Maps* ke dalam sebuah aplikasi yang dibuat oleh pengembang. Pengembang akan mampu mengakses dan memanipulasi data-data geografis. Ada beberapa kegunaan yang bisa digunakan pengembang, di antaranya adalah menampilkan peta interaktif, menandai lokasi atau rute tertentu, mencari tempat, menghitung jarak tempuh, dan sebagainya [15]. *Google Maps API* ini bisa digunakan apabila pengembang sudah mendapatkan *API Key* dari *Google*. Dalam penggunaannya, *Google Maps* ini selalu berkaitan dengan penggunaan *GPS*.

2.2.10. Rumus *Haversine*

Rumus *Haversine* adalah sebuah rumus matematika yang digunakan untuk menghitung jarak antara dua titik pada permukaan bola, seperti Bumi, berdasarkan koordinat lintang dan bujur dari masing-masing titik. Rumus ini sangat berguna dalam navigasi dan pemetaan, terutama ketika menghitung jarak antara dua lokasi geografis di Bumi.

Rumus *Haversine* diberi nama berdasarkan *haversine function*, yang merupakan fungsi trigonometri setengah sudut. Dalam konteks rumus *Haversine*, rumus ini digunakan untuk menghitung jarak melintang antara dua titik pada permukaan bola [16]. Berikut adalah rumus *Haversine*:

$$a = \sin^2\left(\frac{\Delta\emptyset}{2}\right) + \cos\emptyset_1 \cdot \cos\emptyset_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \tan^{-1} 2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

Persamaan 1. Rumus *Haversine*

Dimana:

$\Delta\emptyset$ adalah selisih lintang (*latitude*) antara dua titik dalam radian,

$\Delta\lambda$ adalah selisih bujur (*longitude*) antara dua titik dalam radian,

$\emptyset_1$ dan $\emptyset_2$ adalah lintang dari masing-masing titik dalam radian,

R adalah jari-jari bola (atau bumi),

d adalah jarak antara dua titik.

2.2.11. Camera

Camera adalah perangkat keras yang digunakan untuk menangkap gambar atau merekam video. *Camera* tidak hanya menjadi sebuah perangkat keras yang berdiri sendiri, namun juga bisa menjadi fitur yang terintegrasi dalam *smartphone*, tablet, laptop, atau perangkat elektronik lainnya. *Camera* juga dapat merujuk pada aplikasi yang memungkinkan pengguna untuk mengambil gambar atau video yang mampu diatur pencahayaan, fokus, mode potret, dan sebagainya lewat perangkat-perangkat yang mengintegrasi *camera* [12].

2.2.11.1. CameraX

CameraX adalah sebuah *library Android* yang disediakan oleh *Google* untuk mempermudah pengembangan aplikasi kamera. Tujuan utama dari *CameraX* adalah menyediakan antarmuka yang konsisten dan mudah digunakan untuk mengakses fitur kamera pada berbagai perangkat *Android* dengan versi sistem operasi yang berbeda [12]. Berikut adalah beberapa kelebihan *CameraX* dalam implementasi tampilan gambarnya dibandingkan dengan *library* lainnya:

1. Abstraksi Perangkat Kamera yang Konsisten: *CameraX* menyediakan abstraksi perangkat kamera yang konsisten di seluruh perangkat *Android*, termasuk perangkat dengan konfigurasi kamera yang kompleks. Hal ini memudahkan pengembang untuk menulis kode yang dapat berjalan di

berbagai perangkat *Android* tanpa memerhatikan perbedaan implementasi kamera pada setiap perangkat.

2. Kesederhanaan Penggunaan: *CameraX* dirancang dengan fokus pada kesederhanaan penggunaan. Dibandingkan dengan *library* lain yang mungkin memiliki *API* yang kompleks dan sulit dipahami, *CameraX* menyediakan *API* yang mudah dipelajari dan dipahami oleh pengembang, bahkan untuk pengembang yang relatif baru dalam pengembangan *Android*.
3. Fitur-fitur Tingkat Tinggi: *CameraX* menyediakan fitur-fitur tingkat tinggi yang mempermudah implementasi tampilan gambarnya. Misalnya, dengan *CameraX*, pengembang dapat dengan mudah membuat tampilan gambarnya dengan menggunakan *PreviewView* yang dioptimalkan untuk menampilkan pratinjau kamera secara *real-time*.
4. Kompatibilitas yang Baik: *CameraX* dirancang untuk kompatibilitas yang baik dengan berbagai perangkat Android. Itu berarti kode yang ditulis dengan menggunakan *CameraX* cenderung dapat berjalan dengan baik pada berbagai perangkat *Android*, termasuk perangkat yang lebih lama.
5. Dukungan *Jetpack*: *CameraX* merupakan bagian dari *Jetpack*, kumpulan *library* dan alat yang disediakan oleh *Google* untuk membantu pengembangan aplikasi *Android*. Dengan menggunakan *CameraX*, pengembang mendapatkan dukungan dan pembaruan terus-menerus dari *Google*, serta integrasi yang baik dengan komponen *Jetpack* lainnya.
6. Pembaharuan dan Pemeliharaan: Sebagai *library* resmi dari *Google*, *CameraX* mendapatkan pembaruan dan perbaikan *bug* secara teratur. Hal ini memastikan bahwa aplikasi pengembang dapat memanfaatkan fitur terbaru dan mendapatkan dukungan untuk masalah yang mungkin muncul.

Secara keseluruhan, *CameraX* merupakan *library* yang kuat dan direkomendasikan untuk implementasi tampilan gambarnya dalam pengembangan aplikasi *Android*. Dengan menyederhanakan dan mengabstraksi kompleksitas pengambilan gambar menggunakan kamera, *CameraX* memudahkan pengembang

untuk mengintegrasikan fungsionalitas kamera dengan mudah dan efisien ke dalam aplikasi mereka.

2.2.12. QR Code

QR Code atau *Quick Response Code* adalah jenis kode matriks dua dimensi yang di dalamnya tersimpan informasi dalam bentuk pola persegi hitam dan putih. *QR Code* diciptakan oleh perusahaan Jepang, *Denso Wave*, pada tahun 1994.

QR Code dapat menyimpan berbagai jenis informasi, seperti teks, *URL*, nomor telepon, informasi kontak, alamat email, detail produk, dan banyak lagi [17]. Saat *QR Code* ini di-*scan* menggunakan perangkat yang memiliki kamera dan aplikasi *QR Code reader*, informasi yang terkandung dalam *QR Code* dapat diakses dengan cepat [18].

Keuntungan dari *QR Code* adalah sebagai berikut.

1. Kapasitas penyimpanan yang besar: *QR Code* memiliki kapasitas penyimpanan hingga beberapa kilobita, tergantung pada ukuran dan versi kode sehingga dianggap mampu menyimpan jumlah informasi yang lebih besar daripada *barcode* tradisional.
2. Kemudahan pembacaan: *QR Code* dapat dengan cepat dan mudah dibaca menggunakan perangkat yang memiliki kamera dan aplikasi *QR Code reader*. Aplikasi akan langsung membaca informasi yang terkandung saat pengguna mengarahkan kamera ke *QR Code*.
3. Fleksibilitas: *QR Code* dapat ditempatkan pada berbagai media fisik, seperti kertas, poster, produk, layar komputer, atau layar *smartphone*.
4. Keamanan dan kesalahan pemulihan: *QR Code* memiliki kemampuan untuk mendeteksi dan memperbaiki kesalahan baca hingga tingkat tertentu. Ini berarti saat adanya kerusakan atau gangguan pada *QR Code*, sebagian besar informasi masih dapat diakses.

2.2.11.1. ZXing

ZXing (dibaca "*Zebra Crossing*") adalah sebuah *library open-source* yang populer untuk *scan* dan *generate QR Code* serta kode bar lainnya [19]. *ZXing* dikembangkan dalam bahasa *Java*, tetapi juga mendukung beberapa bahasa pemrograman lainnya seperti *C++, Python, Ruby*, dan lain-lain.

Library ZXing menyediakan fungsi-fungsi untuk:

1. Membaca *QR Code* dan *Barcode (Scanner)*: *ZXing* dapat digunakan untuk memindai dan membaca berbagai jenis *QR Code* dan *barcode*. *Library* ini menyediakan *API* yang memudahkan pengembang dalam mengintegrasikan fungsi pembacaan kode ke dalam aplikasi mereka.
2. Menghasilkan *QR Code* dan *Barcode (Generate)*: *ZXing* juga memungkinkan pembuatan *QR Code* dan *barcode* dalam berbagai format dan ukuran. Pengembang dapat menghasilkan kode dengan teks, *URL*, nomor telepon, dan informasi lainnya, serta mengatur parameter seperti ukuran, format gambar, dan level pemulihan kesalahan.
3. Mendukung Platform yang Beragam: *ZXing* dapat digunakan dalam berbagai platform dan lingkungan pengembangan.
4. Komunitas yang Aktif: *ZXing* memiliki komunitas pengembang yang aktif, yang berarti adanya dukungan dan pembaruan terus-menerus. Pengembang dapat berkontribusi dalam pengembangan *ZXing* atau mendapatkan bantuan dari komunitas ketika menghadapi masalah atau tantangan dalam penggunaan *library* ini.

2.2.13. QRIS

Teknologi *QRIS* atau *Quick Response Code Indonesian Standard* adalah teknologi yang digunakan toko saat melakukan pembayaran pesanan mereka. Teknologi *QRIS* ini merupakan sistem pembayaran *online* yang bisa digunakan pada berbagai jenis *e-wallet* atau *internet banking*. *QRIS* ini secara resmi dikembangkan oleh Bank Indonesia. *QRIS* telah menjadi salah satu syarat pembayaran *online* di Indonesia [20].

2.2.14. JSON

JSON atau *JavaScript Object Notation* adalah format data ringkas untuk pertukaran dan penyimpanan data. *JSON* Format ini banyak digunakan dalam pengembangan aplikasi *web* dan *mobile*, serta dalam komunikasi antar sistem.

JSON menggunakan sintaksis berbasis teks yang terdiri dari format "*key: value*" yang dihubungkan dengan tanda titik dua (":") dan dipisahkan oleh tanda koma (","). Data dalam *JSON* dapat berupa objek (pasangan nama-nilai yang diapit oleh kurung kurawal), *array* (kumpulan nilai yang diapit oleh kurung siku), *string*, angka, *boolean*, atau *null* [21].

JSON dapat digunakan untuk mengirim data antara aplikasi klien dan server melalui permintaan *HTTP* atau untuk menyimpan data dalam format yang terstruktur di dalam *file*. Dalam pengembangan aplikasi, *JSON* sering digunakan sebagai format data yang umum digunakan dalam *RESTful API*, pertukaran data *AJAX*, penyimpanan konfigurasi, dan banyak lagi.

2.2.15. PHP

PHP (*Hypertext Preprocessor*) adalah bahasa pemrograman *server-side* yang sering digunakan untuk pengembangan aplikasi *web*. *PHP* dirancang untuk membangun aplikasi *web* yang dinamis dan interaktif [22]. Bahasa ini dapat diintegrasikan dengan *HTML*.

Beberapa fitur dan kemampuan *PHP* yang umum digunakan dalam pengembangan aplikasi web adalah sebagai berikut:

1. Pengolahan *Form*: *PHP* mengambil data yang dikirim melalui formulir *HTML* dan memprosesnya di sisi server.
2. Koneksi ke *Database*: *PHP* menyediakan berbagai fungsi dan ekstensi untuk terhubung ke berbagai jenis *database*, seperti *MySQL*, *PostgreSQL*, *MongoDB*, dan lain-lain. Dengan *PHP* mampu menjalankan *query SQL*, mengambil data, memperbarui data, dan melakukan operasi lain terhadap *database*.

3. Pembuatan Halaman Dinamis: Dapat membuat halaman web dinamis yang dapat menyesuaikan konten berdasarkan permintaan pengguna dengan *PHP*.
4. Manajemen *File*: *PHP* memiliki fungsi dan metode yang kuat untuk mengelola *file* di server. Mampu mengunggah *file* dari *form*, membaca dan menulis *file* teks, membuat *directory*, menghapus *file*, dan melakukan operasi *file* lainnya menggunakan *PHP*.
5. Manipulasi *String*: *PHP* memiliki berbagai fungsi untuk memanipulasi *string*, seperti menggabungkan, memotong, mencari, mengganti, dan memformat *string*. Fitur ini memudahkan pengolahan dan manipulasi teks dalam aplikasi *web*.
6. Kerangka Kerja atau *Framework*: *PHP* memiliki *framework* populer, seperti *Laravel*, *Symfony*, *CodeIgniter*, dan lain-lain. Kerangka kerja ini menyediakan struktur dan fitur yang siap pakai untuk mempercepat pengembangan aplikasi *web* dengan *PHP*.

2.2.16. JavaScript

JavaScript adalah bahasa pemrograman tingkat tinggi yang digunakan untuk mengembangkan aplikasi web interaktif. Sebagai bahasa pemrograman yang berjalan di sisi klien (*client-side*), *JavaScript* memungkinkan pengembang web untuk menambahkan berbagai fitur interaktif ke dalam halaman web [23]. Misalnya, validasi formulir, efek animasi, manipulasi elemen *HTML* dan *CSS*, komunikasi dengan server, dan masih banyak lagi.

Salah satu bentuk penggunaan *JavaScript* adalah *AJAX*. *AJAX* adalah teknologi yang memungkinkan halaman web untuk mengirim dan menerima data dari server secara *asynchronous* (tanpa memuat ulang seluruh halaman). Dengan *AJAX*, halaman web dapat memperbarui bagian tertentu tanpa harus mengganti seluruh halaman, yang menghasilkan antarmuka pengguna yang lebih responsif dan nyaman.

Pada awalnya, *AJAX* menggunakan format data *XML* untuk berkomunikasi dengan server. Namun, seiring perkembangan waktu, format data lain seperti

JSON (*JavaScript Object Notation*) lebih sering digunakan karena lebih mudah dipahami dan diolah oleh *JavaScript*.

2.2.17. Java

Java adalah bahasa pemrograman yang populer dan kuat yang digunakan secara luas dalam pengembangan aplikasi perangkat lunak. *Java* dikembangkan oleh *Sun Microsystems* (sekarang menjadi *Oracle Corporation*) pada tahun 1995 dan dirancang dengan prinsip "*Write Once, Run Anywhere*" yang memungkinkan kode *Java* dapat dijalankan di berbagai platform yang mendukung *Java*, termasuk *Windows, macOS, Linux*, dan perangkat seluler [24].

Berikut adalah beberapa fitur dan konsep yang terkait dengan *Java*:

1. *Platform-Independent*: Kode *Java* dikompilasi menjadi *bytecode* yang dapat dijalankan pada *Java Virtual Machine (JVM)*. *JVM* bertindak sebagai mesin virtual yang mengeksekusi *bytecode Java* di berbagai platform. Ini memungkinkan aplikasi *Java* untuk berjalan secara konsisten di berbagai sistem operasi tanpa perlu melakukan perubahan kode.
2. *Object-Oriented Programming (OOP): Java* merupakan bahasa pemrograman yang sepenuhnya berorientasi objek. Dalam *Java*, semuanya adalah objek, termasuk kelas, objek, metode, dan variabel. Konsep *OOP* seperti enkapsulasi, pewarisan, dan polimorfisme dapat diterapkan dengan mudah dalam pengembangan aplikasi *Java*.
3. Keamanan: *Java* memiliki fitur keamanan yang kuat. *Sandboxing* dan mekanisme keamanan yang terintegrasi dalam *JVM* memastikan bahwa aplikasi *Java* berjalan dengan aman dan tidak mengancam sistem yang menjalankannya. *Java* juga mendukung pembuatan aplikasi dengan sertifikat digital untuk memastikan integritas dan otentikasi kode.
4. *Rich Standard Library*: Java menyediakan sebuah *Standard Library* yang kaya dan komprehensif. *Library* ini mencakup

berbagai kelas dan metode yang memudahkan pengembang dalam mengimplementasikan fungsionalitas yang umum digunakan, seperti manipulasi *string*, operasi *I/O*, pengelolaan *thread*, koneksi jaringan, pengolahan data, dan masih banyak lagi.

5. *Multi-Threading*: *Java* memiliki dukungan yang kuat untuk pemrograman *multi-threading*, yang memungkinkan eksekusi simultan dari beberapa tugas dalam satu aplikasi. Hal ini memungkinkan pengembang untuk membuat aplikasi yang responsif, melakukan operasi paralel, dan memanfaatkan kemampuan pemrosesan yang lebih baik dalam lingkungan *multi-core*.

2.2.18. Database

Database adalah suatu sistem yang digunakan untuk menyimpan, mengelola, dan mengorganisir data dalam struktur yang terstruktur dan efisien. *Database* memungkinkan pengguna untuk menyimpan dan mengakses data dengan mudah serta melakukan operasi seperti pengambilan, penyimpanan, pembaruan, dan penghapusan data (*CRUD*) [25].

Struktur database mengacu pada cara data disusun dan diatur dalam suatu sistem *database*. Berikut ini beberapa komponen penting dalam struktur *database*:

1. Tabel: Tabel adalah unit dasar dalam struktur *database*. Tabel terdiri dari baris dan kolom. Setiap kolom mewakili atribut atau jenis data yang disimpan, sedangkan setiap baris mewakili entitas atau rekaman data. Tabel harus memiliki kunci primer yang unik untuk mengidentifikasi setiap baris secara unik.
2. Kolom: Kolom adalah bagian dari tabel yang merepresentasikan atribut atau jenis data. Setiap kolom memiliki nama dan tipe data yang sesuai, seperti teks, angka, tanggal, atau *boolean*. Kolom juga bisa memiliki batasan, seperti batasan *null* atau kunci unik (*unique key*), yang mengatur keunikan atau keharusan nilai dalam kolom tersebut.
3. Kunci Primer (*Primary Key*): Kunci primer adalah kolom atau kelompok kolom dalam sebuah tabel yang secara unik mengidentifikasi setiap baris

dalam tabel tersebut. Kunci primer memastikan keunikan data dan memungkinkan akses yang cepat ke data tertentu. Biasanya, kunci primer diwakili oleh kolom dengan tipe data unik seperti *ID* atau nomor unik.

4. Kunci Asing (*Foreign Key*): Kunci asing adalah kolom atau kelompok kolom dalam sebuah tabel yang menghubungkan tabel dengan tabel lain dalam hubungan. Kunci asing digunakan untuk menciptakan relasi antar tabel dan mengintegrasikan data dari tabel yang berbeda. Kunci asing berhubungan dengan kunci primer dalam tabel lain, membentuk hubungan antara entitas.
5. Indeks: Indeks adalah struktur data yang digunakan untuk meningkatkan kecepatan pencarian dan pengaksesan data dalam tabel. Indeks dibangun di atas satu atau beberapa kolom dalam tabel dan memungkinkan pencarian data berdasarkan nilai kolom yang diindeks. Indeks sering digunakan pada kolom yang sering digunakan dalam operasi pencarian, seperti kunci primer atau kolom yang sering digunakan dalam klausul *WHERE*.
6. Skema: Skema *database* adalah struktur keseluruhan yang mendefinisikan tabel, hubungan antar tabel, dan properti lainnya dalam *database*. Skema menggambarkan entitas, atribut, dan hubungan dalam *database*, serta aturan dan batasan yang berlaku. Skema biasanya didefinisikan menggunakan bahasa seperti *SQL (Structured Query Language)* atau melalui alat manajemen *database*.

2.2.16.1. MySQL

MySQL merupakan salah satu *Relational Database Management Sistem (RDBMS)* [26]. *MySQL* dikembangkan oleh perusahaan *Oracle Corporation*. *MySQL* bersifat *open source*, artinya dapat digunakan secara gratis, dan tersedia dalam berbagai versi dan edisi.

Berikut ini adalah beberapa kemampuan utama dari *MySQL*:

1. Penyimpanan dan Pengelolaan Data: *MySQL* dapat digunakan untuk menyimpan dan mengelola berbagai jenis data, termasuk teks, angka,

tanggal, gambar, dan lainnya. *MySQL* mendukung berbagai tipe data seperti *INTEGER, VARCHAR, DATE, BLOB*, dan lainnya.

2. Bahasa Pemrograman dan *API*: *MySQL* mendukung berbagai bahasa pemrograman seperti *PHP, Java, Python, C++,* dan sebagainya. *MySQL* menyediakan *driver* dan *API* yang komprehensif untuk berinteraksi dengan basis data menggunakan bahasa pemrograman yang didukung.
3. *Query Language*: *MySQL* menggunakan bahasa *query SQL (Structured Query Language)* sebagai antarmuka untuk mengakses dan memanipulasi data. *SQL* adalah bahasa standar yang digunakan dalam sistem basis data relasional, dan *MySQL* menyediakan fitur lengkap dari *SQL*, termasuk *SELECT, INSERT, UPDATE, DELETE, JOIN*, dan sebagainya.
4. Kecepatan dan Kinerja: *MySQL* dirancang untuk memiliki kinerja yang cepat dan efisien. *MySQL* menggunakan berbagai teknik dan algoritma optimisasi untuk menjalankan query dengan cepat, seperti *indexing*, *caching*, dan pengoptimalan *query*. *MySQL* juga mendukung penggunaan indeks yang cerdas untuk meningkatkan kinerja pencarian data.
5. Keamanan: *MySQL* menyediakan fitur keamanan yang kuat untuk melindungi data. Ini termasuk otentikasi pengguna yang aman, izin akses yang fleksibel, enkripsi data, dan dukungan *SSL/TLS* untuk komunikasi yang aman antara klien dan server.
6. Replikasi dan Skalabilitas: *MySQL* mendukung fitur replikasi yang memungkinkan replikasi data ke beberapa server. Replikasi ini meningkatkan ketersediaan dan ketahanan sistem basis data. *MySQL* juga mendukung partisi tabel yang memungkinkan pengguna untuk mempartisi data menjadi beberapa bagian untuk meningkatkan skalabilitas.
7. Dukungan Transaksi: *MySQL* mendukung transaksi *ACID (Atomicity, Consistency, Isolation, Durability)* untuk memastikan integritas dan konsistensi data.
8. Pemulihan dan *Backup*: *MySQL* memiliki dukungan untuk pemulihan setelah kegagalan sistem dan juga menyediakan alat untuk *backup* dan pemulihan data.

9. Pembuatan Laporan dan Analisis: *MySQL* menyediakan fitur untuk menghasilkan laporan dan melakukan analisis data. Dengan menggunakan perintah *SQL* yang kompleks, pengguna dapat melakukan penggalian data, pengelompokan data, dan perhitungan agregat untuk menghasilkan informasi yang berharga dari basis data.
10. Dukungan Komunitas: *MySQL* memiliki komunitas pengguna yang besar dan aktif, yang menyediakan sumber daya, forum diskusi, dan dokumentasi resmi. Pengguna *MySQL* dapat dengan mudah mendapatkan dukungan dan berbagi pengetahuan dengan anggota komunitas lainnya.

2.2.19. ERD

ERD (Entity Relationship Diagram) adalah sebuah model konseptual yang digunakan untuk menggambarkan hubungan antara entitas dalam basis data. *ERD* menggambarkan struktur basis data dalam bentuk diagram yang menggunakan entitas, atribut, dan hubungan antara entitas sebagai komponen utamanya [27].

Dalam *ERD*, entitas mewakili objek atau konsep dalam dunia nyata yang ingin direpresentasikan dalam basis data. Setiap entitas memiliki atribut yang menggambarkan karakteristik atau properti dari entitas tersebut.

ERD menggunakan simbol-simbol grafis seperti persegi panjang (untuk entitas), lingkaran (untuk atribut), dan garis-garis (untuk hubungan) untuk menggambarkan komponen-komponen tersebut. *ERD* juga dapat menggunakan notasi tambahan seperti *kardinalitas* (untuk menggambarkan jumlah relasi antara entitas) dan atribut turunan (yang dihitung dari atribut lainnya) untuk memperkaya representasi model konseptual [28].

2.2.20. Diagram Relasi

Diagram Relasi atau *Relational Schema Diagram* adalah representasi grafis dari skema basis data relasional. Diagram ini menggambarkan tabel-tabel dalam basis data, kolom-kolom yang ada dalam setiap tabel, serta hubungan antara tabel-tabel tersebut melalui kunci primer dan kunci asing.

Diagram Relasi memperlihatkan struktur basis data relasional secara visual, dengan tabel-tabel yang diwakili oleh persegi panjang dan kolom-kolom

yang diwakili oleh nama kolom di dalam tabel. Hubungan antara tabel-tabel diindikasikan dengan menggunakan kunci primer *(primary key)* dan kunci asing *(foreign key)* [29], [30]. Kunci primer adalah kolom atau kombinasi kolom yang secara unik mengidentifikasi setiap baris dalam sebuah tabel. Kunci asing adalah kolom yang mengacu pada kunci primer di tabel lain, menciptakan hubungan antara tabel-tabel dalam basis data.

Selain itu, Diagram Relasi juga dapat menunjukkan *kardinalitas* hubungan antara entitas. *Kardinalitas* menggambarkan jumlah entitas yang terhubung melalui hubungan tertentu. Misalnya, hubungan satu-ke-banyak antara dua tabel dapat ditunjukkan dengan tanda "1" pada satu sisi hubungan dan tanda "N" pada sisi lainnya, menunjukkan bahwa satu entitas di satu sisi hubungan dapat berhubungan dengan banyak entitas di sisi lainnya.

2.2.21. Use Case Diagram

Use Case Diagram adalah jenis diagram yang digunakan dalam rekayasa perangkat lunak untuk menggambarkan interaksi antara aktor (*user* atau sistem eksternal) dengan sistem yang sedang direncanakan atau dikembangkan. Diagram *Use Case* digunakan untuk mengidentifikasi, menganalisis, dan menggambarkan fungsi-fungsi sistem dari perspektif pengguna.

Use Case Diagram menggambarkan skenario-skenario atau kasus penggunaan yang menjelaskan bagaimana pengguna atau aktor menggunakan sistem untuk mencapai tujuan tertentu. Setiap kasus penggunaan (*use case*) menggambarkan sebuah fungsi atau tindakan tertentu yang dapat dilakukan oleh pengguna atau aktor dalam sistem.

Komponen utama dalam *Use Case Diagram* adalah sebagai berikut:

1. Aktor (*Actor*): Aktor mewakili pengguna sistem atau entitas eksternal lainnya yang berinteraksi dengan sistem. Aktor dapat berupa pengguna manusia, perangkat keras eksternal, sistem lain, atau entitas bisnis lainnya.
2. *Use Case*: *Use Case* mewakili sebuah fungsi atau tindakan yang dapat dilakukan oleh pengguna atau aktor dalam sistem. *Use Case*

menggambarkan interaksi antara aktor dan sistem, dan menjelaskan bagaimana sistem merespons aksi dari pengguna atau aktor.

3. Asosiasi (*Association*): Asosiasi menghubungkan aktor dengan *use case* yang terkait. Ini menunjukkan bahwa aktor terlibat dalam eksekusi atau penggunaan *use case* tertentu.
4. Ekstensi (*Extension*): Ekstensi menggambarkan skenario alternatif atau perubahan dalam kasus penggunaan utama. Ini menggambarkan kondisi tambahan atau variasi yang mungkin terjadi dalam eksekusi *use case*.

2.2.22. Activity Diagram

Diagram Aktivitas *(Activity Diagram)* adalah jenis diagram dalam rekayasa perangkat lunak yang digunakan untuk menggambarkan aliran kerja atau urutan aktivitas dalam suatu proses bisnis, sistem, atau *use case*. *Activity Diagram* memberikan visualisasi yang jelas tentang bagaimana aktivitas-aktivitas berhubungan satu sama lain, bagaimana aliran kontrol dilakukan, serta bagaimana pengambilan keputusan dan percabangan terjadi dalam proses tersebut.

Activity Diagram menggunakan simbol-simbol grafis untuk menggambarkan aktivitas, penghubung, keputusan, garis waktu, dan kondisi. Beberapa komponen penting dalam *Activity Diagram* meliputi [30]:

1. *Node* Aktivitas (*Activity Node*): *Activity Node* mewakili suatu tindakan atau aktivitas dalam proses. *Node* aktivitas dapat berupa tindakan (*action*), pengiriman pesan (*send message*), penerimaan pesan (*receive message*), atau pemilihan keputusan (*decision*).
2. Penghubung (*Connector*): Penghubung digunakan untuk menghubungkan *node-node* aktivitas dalam urutan yang logis. Penghubung dapat berupa panah atau garis yang menunjukkan aliran kontrol dari satu aktivitas ke aktivitas berikutnya.
3. Keputusan (*Decision*): Keputusan digunakan untuk memodelkan kondisi atau percabangan dalam proses. Keputusan dapat mengarahkan aliran kontrol ke aktivitas yang berbeda tergantung pada kondisi yang ada.

4. Garis Waktu (*Swimlane*): Garis waktu digunakan untuk mengelompokkan aktivitas-aktivitas dalam beberapa kelompok atau pemangku kepentingan yang berbeda. Garis waktu biasanya mewakili peran atau aktor dalam sistem.

2.2.23. Sequence Diagram

Diagram urutan (*Sequence Diagram*) adalah jenis diagram dalam rekayasa perangkat lunak yang digunakan untuk menggambarkan interaksi antara objek-objek dalam sebuah sistem atau proses. Diagram urutan menunjukkan urutan pesan yang dikirim antara objek-objek selama waktu tertentu dan menggambarkan bagaimana objek-objek berinteraksi satu sama lain.

Komponen-komponen utama dalam *sequence diagram* meliputi [30]:

1. Objek (*Object*): Objek mewakili entitas yang berpartisipasi dalam interaksi. Objek dapat berupa kelas, *instance* kelas, atau komponen sistem lainnya. Setiap objek memiliki nama yang mengidentifikasinya.
2. Pesan (*Message*): Pesan mewakili komunikasi antara objek-objek. Pesan dapat berupa pesan terkirim (*message sent*) yang dikirimkan oleh satu objek ke objek lain, atau pesan diterima (*message received*) yang diterima oleh objek dari objek lain. Pesan juga dapat mengandung informasi tambahan seperti argumen atau parameter.
3. Garis Waktu (*Lifeline*): Garis waktu adalah garis vertikal yang terhubung dengan objek dan menggambarkan waktu hidup objek selama interaksi. Garis waktu menunjukkan saat objek dibuat (muncul) dan saat objek dihancurkan (hilang).
4. Pengendali (*Control Flow*): Pengendali adalah garis horizontal yang menghubungkan pesan dan menggambarkan urutan pesan antara objek-objek. Pengendali menunjukkan aliran kontrol dari satu objek ke objek lain.
5. Pesan Panggilan Balik (*Callback Message*): Pesan panggilan balik adalah pesan yang dikirim oleh objek setelah menerima suatu peristiwa tertentu. Pesan panggilan balik digunakan untuk menggambarkan

tanggapan atau tindakan yang dilakukan oleh objek setelah menerima pesan dari objek lain.

2.2.24. Class Diagram

Class diagram adalah salah satu jenis diagram dalam *UML* (*Unified Modeling Language*) yang digunakan untuk menggambarkan struktur kelas dalam suatu sistem atau aplikasi. Diagram ini membantu dalam analisis dan perencanaan sistem secara visual dengan menunjukkan hubungan antar kelas, atribut, dan metode yang dimiliki oleh setiap kelas.

Notasi pada *class diagram* terdiri dari simbol-simbol yang mewakili elemen-elemen penting dari kelas, seperti [30]:

1. Nama Kelas:

   Nama kelas ditulis di atas kotak yang mewakili kelas. Contohnya, kita memiliki kelas "*Person*".

2. Atribut:

   Atribut adalah variabel atau properti yang dimiliki oleh kelas. Notasi atribut ditulis di dalam kotak kelas dan biasanya diawali dengan simbol visibilitas, seperti "+" untuk *public*, "-" untuk *private*, "#" untuk *protected*, atau "~" untuk *package-private*. Misalnya, kelas "*Person*" memiliki atribut "*name*" dan "*age*".

3. Metode:

   Metode adalah fungsi atau operasi yang dapat dilakukan oleh kelas. Notasi metode juga ditulis di dalam kotak kelas dan diawali dengan simbol visibilitas. Misalnya, kelas "Person" memiliki metode "*getName*()" dan "*getAge*()".

4. Hubungan antar Kelas:

   Hubungan antar kelas diwakili oleh tanda panah yang menghubungkan kelas-kelas yang terlibat. Beberapa jenis hubungan pada *class diagram* antara lain:

   - Asosiasi (*Association*): Hubungan umum yang menunjukkan bahwa dua atau lebih kelas memiliki keterkaitan.

- Pewarisan (*Inheritance*): Hubungan keturunan antara kelas "*parent*" (*superclass*) dan "*child*" (*subclass*).
- Agregasi (*Aggregation*): Hubungan yang menunjukkan bahwa satu kelas (kelas utama) berisi objek dari kelas lain (kelas bagian).
- Komposisi (*Composition*): Hubungan yang menunjukkan bahwa satu kelas (kelas utama) memiliki objek dari kelas lain (kelas bagian) yang terikat secara eksklusif.
- Ketergantungan (*Dependency*): Hubungan yang menunjukkan bahwa satu kelas menggunakan layanan dari kelas lain tanpa memiliki ketergantungan yang kuat.