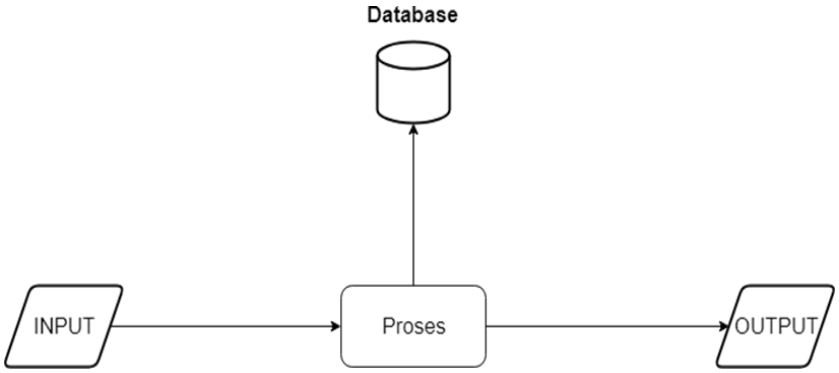
II. TINJAUAN PUSTAKA

2.1 Konsep Dasar Sistem Informasi

Sistem informasi adalah sekumpulan elemen yang saling terkait dan bekerja sebagai satu kesatuan untuk mengintegrasikan, memproses, menyimpan, serta mendistribusikan informasi. Sistem informasi adalah sebuah sistem dalam organisasi yang menyelaraskan kebutuhan pemrosesan transaksi harian, mendukung fungsi operasional organisasi yang bersifat manajerial, serta kegiatan strategis, sehingga dapat menyediakan laporan yang diperlukan bagi pihak-pihak eksternal tertentu. (Sutabri , 2012).



Gambar 1. Ilustrasi Proses Kerja Sebuah Sistem (Sumber: Sutabri, 2012).

Sistem informasi terdiri dari tiga komponen utama yang membentuk suatu siklus, yaitu:

1. Input

   Input merupakan data yang dimasukkan ke dalam sistem informasi. Input ini mencakup metode dan media yang digunakan untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen - dokumen dasar.
2. Proses

   Proses adalah gabungan dari prosedur, logika, dan model yang berfungsi untuk memanipulasi data input dan data yang tersimpan dalam basis data, sesuai dengan metode tertentu, guna menghasilkan output yang diharapkan.

3. Output

   Hasil dari sistem informasi adalah output berupa informasi berkualitas dan dokumentasi yang bermanfaat bagi semua tingkat manajemen serta seluruh pengguna sistem.

2.2 Sistem Presensi

Presensi merupakan salah satu aspek krusial dalam pengelolaan sumber daya manusia (SDM) di berbagai organisasi. Melalui pencatatan kehadiran, manajemen dapat memantau disiplin pegawai serta menentukan evaluasi kinerja secara lebih objektif. Sistem presensi yang baik tidak hanya membantu organisasi dalam mengelola kehadiran pegawai, tetapi juga berdampak langsung pada produktivitas dan efisiensi operasional perusahaan.

Sistem presensi manual, dilakukan oleh pegawai dengan menandatangani buku hadir saat tiba dan pulang kerja. Namun, sistem ini memiliki banyak kelemahan, termasuk kemungkinan manipulasi data dan ketidakakuratan pencatatan. Seiring dengan perkembangan teknologi, sistem presensi manual mulai ditinggalkan dan digantikan dengan sistem presensi berbasis teknologi, seperti mesin absensi sidik jari (*fingerprint*), kartu *RFID*, dan sistem berbasis *web* serta *mobile*.

Sistem presensi modern yang berbasis teknologi menawarkan berbagai keuntungan, di antaranya kecepatan pencatatan, akurasi data, serta kemudahan akses bagi manajemen untuk memantau kehadiran pegawai *(Rokhayah et al. 2021)*. Salah satu inovasi yang paling banyak diterapkan adalah sistem presensi berbasis *fingerprint*, di mana kehadiran pegawai dicatat melalui verifikasi biometrik sidik jari. Sistem ini secara signifikan mengurangi peluang manipulasi data kehadiran, karena sidik jari bersifat unik dan sulit dipalsukan.

Akan tetapi sistem *fingerprint* juga memiliki kendala salah satu kendala yang sering dihadapi adalah keterbatasan dalam akses data secara *real-time*, terutama ketika tidak ada integrasi dengan jaringan *internet*. Hal ini dapat menyebabkan keterlambatan dalam pembaruan informasi kehadiran, sehingga mengurangi efektivitas pengawasan.

2.2 Teknologi Web dan Mobile

Teknologi web dan mobile merupakan dua pilar utama dalam pengembangan sistem informasi modern. Penggunaan kedua teknologi ini memungkinkan perusahaan untuk memantau dan mencatat kehadiran karyawan secara efisien dan *real-time*, serta memudahkan aksesibilitas bagi karyawan dan manajemen di berbagai lokasi.

2.2.1 Teknologi Web

Teknologi web memungkinkan pengguna mengakses melalui browser tanpa memerlukan instalasi khusus. Dalam sistem presensi karyawan, teknologi web digunakan untuk memantau dan mengelola kehadiran karyawan secara terpusat, dengan data yang disimpan secara aman di server. Teknologi inti yang digunakan dalam aplikasi berbasis web mencakup *HTML, CSS, dan JavaScript*, yang memungkinkan pembuatan antarmuka pengguna yang interaktif dan responsif. Penggunaan teknologi ini memungkinkan pengelolaan kehadiran secara *real-time* melalui *dashboard* berbasis *web*, yang menyediakan informasi langsung bagi manajemen untuk mengambil keputusan secara cepat dan akurat *(Bharathy et al. 2021).*

Keunggulan lain dari teknologi *web* adalah kemampuannya dalam memperbarui sistem secara terpusat. Pembaruan dapat dilakukan di sisi *server* tanpa melibatkan pengguna, memastikan bahwa sistem selalu berada pada versi terbaru, serta menjaga keamanan dan kinerja sistem.

2.2.2 Teknologi Mobile

Teknologi mobile menawarkan fleksibilitas yang lebih besar, karena memungkinkan karyawan untuk mencatat kehadiran mereka menggunakan perangkat seluler. Aplikasi presensi berbasis *mobile* dapat memanfaatkan berbagai fitur, seperti *GPS* untuk memvalidasi lokasi kehadiran, dan pengenalan wajah untuk memastikan keakuratan data kehadiran.

Menurut Penelitian yang dilakukan oleh *(Sunaryono et al. 2021)* menunjukkan bagaimana teknologi *mobile* berbasis Android digunakan untuk sistem presensi dengan pengenalan wajah. Teknologi ini memungkinkan karyawan mencatat kehadiran mereka secara otomatis dan akurat menggunakan kamera perangkat seluler. Data kehadiran kemudian diintegrasikan dengan sistem berbasis web yang memungkinkan manajemen memantau kehadiran secara real-time.

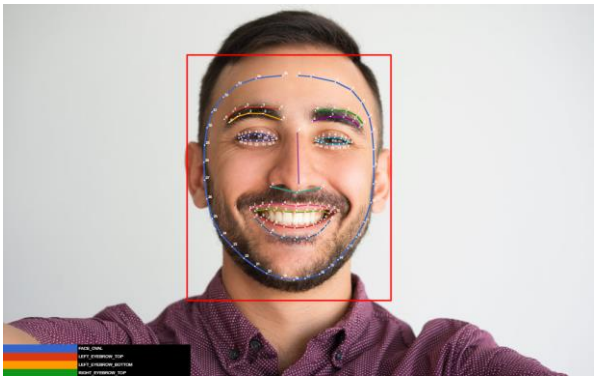
2.2.3 Integrasi Teknologi Web dan Mobile dalam Sistem Presensi

Integrasi teknologi web dan mobile memungkinkan perusahaan untuk memanfaatkan keunggulan kedua platform ini secara bersamaan. Karyawan dapat mencatat kehadiran melalui aplikasi *mobile*, sementara manajemen dapat memantau kehadiran melalui *dashboard* secara real-time.

(Devaprakash et al. 2020) dalam penelitiannya mengenai sistem monitoring kehadiran terpusat, menguraikan bagaimana teknologi *Cloud* dan *IoT* dapat digunakan untuk mengintegrasikan aplikasi *web* dan *mobile* dalam satu sistem presensi yang efisien. Sistem ini memungkinkan pencatatan

kehadiran yang aman dan akurat melalui teknologi *mobile*, dengan data yang tersimpan dan dikelola secara terpusat di *server* berbasis *web*.

2.3 Face Recognition



Gambar 2. Deteksi Wajah

Pengenalan wajah merupakan teknologi *biometrik* yang berfungsi untuk mengidentifikasi atau memverifikasi identitas seseorang dari gambar digital. Teknologi ini bekerja dengan membandingkan ciri-ciri wajah pada gambar yang dianalisis dengan ciri-ciri yang telah tersimpan dalam *database*. Proses ini melibatkan analisis berbagai elemen wajah, seperti bentuk dan tekstur, untuk mencari kesamaan yang akan menentukan kecocokan identitas. Proses dalam pengenalan wajah melibatkan segmentasi area wajah dari latar belakang pada citra masukan. Proses ini dilakukan dengan memeriksa setiap citra berdasarkan parameter yang telah ditentukan untuk mengidentifikasi keberadaan ciri-ciri wajah. Seperti yang diungkapkan oleh (Mohammad, 2020), teknologi ini telah diimplementasikan dalam berbagai bidang untuk memungkinkan pemantauan dan pengawasan, menyoroti potensi dan batasan teknologi pengenalan wajah.

Berbeda dari teknologi *biometrik* lain, teknologi pengenalan wajah unik karena dapat mengidentifikasi banyak orang secara bersamaan. Hal ini sangat berguna dalam situasi seperti pencarian orang hilang. Menurut (Nasution 2020) ada 3 tahapan dalam melakukan Pengenalan Wajah yaitu:

1. *Face detection* : pada tahap ini mendeteksi ada tidaknya wajah pada gambar atau video yang dimasukkan.
2. *Feature Extraction*: setelah wajah terdeteksi, ekstraksi fitur dilakukan untuk mendapatkan fitur wajah.
3. *Face Recognition*: tahap terakhir adalah pengenalan wajah dengan membandingkan wajah yang memiliki karakteristik yang diketahui dengan wajah yang ada di *database*.

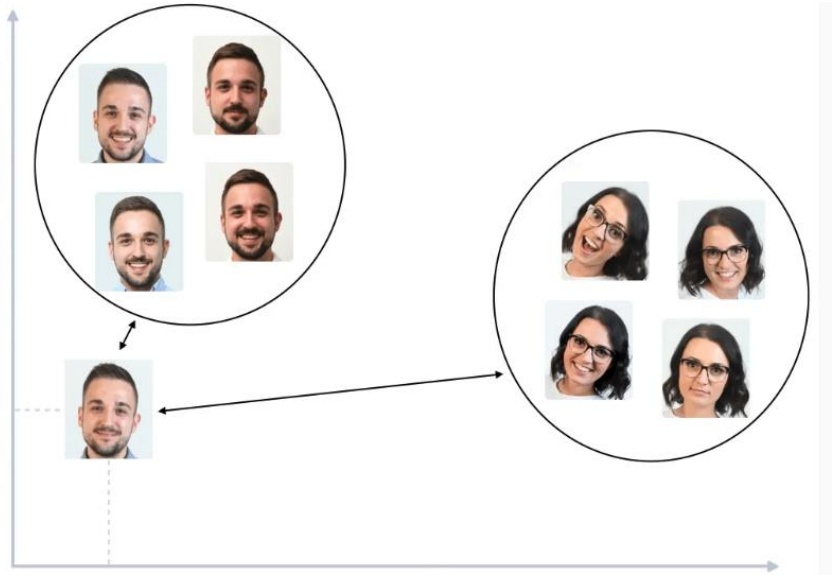
2.4 *FaceNet*

FaceNet adalah model teknologi yang menggunakan gambar untuk mengubah wajah menjadi vektor 128D *Euclidean* (*Embedding*). Model ini dilatih untuk membedakan kemiripan atau perbedaan wajah, memungkinkan pengelompokan, verifikasi, dan pengenalan wajah secara efektif melalui ruang vektor yang dihasilkan. Selain itu, penggunaan embedding 128D dalam FaceNet memungkinkan pengelompokan dan identifikasi wajah yang kuat, bahkan dalam skenario dengan variasi pencahayaan, pose, dan ekspresi yang besar (Sydor dkk. 2024)

Embedding dibuat ketika model dilatih untuk mengenali wajah dengan menggunakan foto wajah sebagai sampel. Untuk membandingkan dua gambar, embedding dari kedua foto wajah dibuat terpisah, kemudian menggunakan rumus tertentu untuk menghitung jarak. Jarak terkecil menunjukkan wajah yang serupa, sedangkan jarak tertinggi menunjukkan wajah yang berbeda.

FaceNet merupakan jaringan saraf dalam yang digunakan untuk mengekstrak fitur dari gambar wajah seseorang. *FaceNet* mengubah gambar wajah menjadi *vektor* 128 angka (*embedding*) yang mewakili fitur paling penting dari wajah tersebut. *Vektor* ini dapat dianggap sebagai titik dalam sistem koordinat *Cartesian*, di mana semua informasi penting dari gambar wajah dikompresi menjadi representasi numerik yang efisien. Seperti yang dijelaskan oleh Arsfrutura (2019), *FaceNet* mengambil gambar wajah sebagai input dan menghasilkan *vektor* 128 angka yang menyimpan fitur-fitur kunci wajah, memungkinkan pengenalan dan perbandingan wajah yang akurat.

Salah satu metode yang dapat digunakan untuk mengenali seseorang dalam gambar yang belum pernah dilihat sebelumnya adalah dengan menghitung embedding-nya dan membandingkan jarak kemiripannya dengan embedding gambar orang yang sudah dikenal. Jika embedding wajah pada gambar tersebut cukup mirip dengan embedding orang A, maka dapat disimpulkan bahwa gambar tersebut mengandung wajah orang A. Untuk melatih *FaceNet*, diperlukan banyak gambar wajah. Sebagai contoh sederhana, misalkan kita memiliki dua gambar wajah orang. Prinsip yang sama dapat diterapkan bahkan jika kita memiliki ribuan gambar dari orang yang berbeda.



Gambar 3. Menganalisa Kemiripan Wajah

Pada awal pelatihan, *FaceNet* menghasilkan vektor acak untuk setiap gambar, yang berarti gambar-gambar tersebut akan terdistribusi secara acak saat diplot. FaceNet belajar melalui proses berikut:

1. Memilih gambar sesuai dengan sampel.
2. Secara acak memilih gambar orang yang mirip dengan sampel gambar. (Contoh Positif).
3. Secara acak memilih gambar seseorang yang berbeda dari sampel gambar. (Contoh Negatif).
4. Menyesuaikan parameter jaringan FaceNet sehingga contoh positif lebih dekat ke sampel gambar dibandingkan dengan contoh negatif.

FaceNet perlu mengidentifikasi fitur utama wajah seseorang yang membedakannya dari wajah yang berbeda. *FaceNet* mencoba banyak kombinasi berbeda selama training hingga menemukan yang paling cocok dengan sampel gambar.

$$f\left(\text{  }\right) = \begin{pmatrix} 0.112 \\ 0.067 \\ 0.091 \\ 0.129 \\ 0.002 \\ 0.012 \\ 0.175 \\ \vdots \\ 0.023 \end{pmatrix}$$

Gambar 4. Mengubah gambar jadi angka 128 vektor

FaceNet berfungsi sebagai model yang mengambil gambar sebagai input dan menghasilkan vektor numerik (*face embedding*) sebagai output. Selanjutnya, FaceNet membandingkan wajah yang tertangkap kamera dengan gambar sampel yang paling mirip.

2.4 *Global positioning system* (GPS)

Gambar 5. Global positioning system (GPS) Satelit

(Sumber: Nasa)

Global Positioning System (GPS) merupakan sistem navigasi berbasis satelit yang memungkinkan penentuan lokasi secara presisi di permukaan bumi. GPS pertama kali dikembangkan oleh Departemen Pertahanan Amerika Serikat untuk aplikasi militer. Menurut *(Kaplan & Hegarty, 2006:3)* dalam bukunya *"Understanding GPS: Principles and Applications"*, GPS bekerja dengan mengandalkan sinyal dari minimal empat satelit yang mengelilingi bumi. Penerima GPS pada perangkat pengguna menangkap sinyal ini dan menggunakan perhitungan jarak berdasarkan waktu tempuh sinyal untuk menentukan lokasi pengguna dengan akurasi yang sangat tinggi. Sinyal-sinyal dari satelit ini berisi

data waktu dan lokasi satelit saat sinyal dikirim, memungkinkan penerima untuk menghitung koordinat geografis berupa lintang, bujur, dan ketinggian.

Prinsip dasar kerja GPS adalah metode *triangulasi*, di mana jarak antara penerima dan satelit dihitung untuk menentukan lokasi pengguna secara tiga dimensi. Setiap satelit GPS dilengkapi dengan jam atom yang sangat akurat, dan perhitungan waktu tempuh sinyal menjadi elemen kunci dalam menentukan jarak. Koreksi waktu juga diperlukan mengingat satelit GPS bergerak dengan kecepatan tinggi dan terpengaruh oleh gravitasi bumi, yang sesuai dengan teori relativitas Einstein.

GPS menawarkan sejumlah keunggulan, termasuk akurasi yang dapat mencapai beberapa meter, kemudahan penggunaan, dan kemampuan bekerja di seluruh dunia. Teknologi GPS telah menjadi elemen penting dalam aplikasi-aplikasi modern seperti pelacakan kendaraan, navigasi kapal, serta sistem pemantauan dan pencatatan kehadiran berbasis lokasi.

Dengan akurasi yang tinggi dan penerapan yang luas, GPS kini menjadi komponen penting dalam berbagai sistem modern, dari peta digital hingga sistem manajemen logistik, serta menjadi tulang punggung dalam layanan berbasis lokasi.

2.5 React js

React.js**,** sering disebut sebagai *React*, adalah *library JavaScript* yang dikembangkan oleh Facebook pada tahun 2013. React memfasilitasi pembangunan antarmuka pengguna (*user interface*/*UI*) secara efisien, terutama untuk aplikasi web yang interaktif dan dinamis. Kepopulerannya di dunia pengembangan web modern tidak lepas dari kemampuannya menangani tampilan kompleks dengan kinerja yang tinggi.

Menurut dokumentasi resmi React.js, beberapa fitur utama yang menjadikan React populer dan relevan dalam pen gembangan aplikasi web adalah sebagai berikut:

1) *Component-Based Architecture*

   React memungkinkan antarmuka pengguna dibangun dengan memecahnya menjadi komponen-komponen kecil yang dapat digunakan kembali. Setiap komponen bertanggung jawab atas pengelolaan status (state) dan logika tampilannya sendiri, sehingga memudahkan pengembangan yang modular dan terorganisir.

2) *Virtual DOM*

   React menggunakan Virtual DOM untuk meningkatkan kinerja. Virtual DOM adalah representasi ringan dari DOM asli. Ketika ada perubahan

pada state atau props, React hanya akan memperbaiki elemen yang berubah tanpa merender ulang seluruh halaman

3) *Sintaks Deklarasi (Declarative Syntax)*

   Dengan sintaks deklaratif, pengembang hanya perlu mendeklarasikan tampilan yang diinginkan, dan React memastikan bahwa UI selalu konsisten dengan status aplikasi. Hal ini membuat kode lebih mudah dibaca dan dipelihara dibandingkan pendekatan imperatif tradisional.

4) *Hooks*

   Fitur Hooks di React memungkinkan penggunaan state dan fitur lainnya tanpa perlu menulis kelas. Hooks memperkenalkan cara baru untuk mengelola logika komponen secara lebih bersih dan terstruktur. Salah satu hook yang paling sering digunakan adalah useState untuk mengelola state dan useEffect untuk menangani efek samping, seperti mengambil data dari API.

2.6 Next Js

Next.js adalah *framework* berbasis React yang memberikan kemudahan bagi pengembang dalam membangun aplikasi web dengan performa tinggi dan optimal. *Framework* ini dirancang untuk meningkatkan produktivitas dan pengalaman pengembangan dengan menghadirkan berbagai fitur unggulan yang mempermudah pengelolaan frontend dan backend dalam satu ekosistem.

Menurut dokumentasi resmi *Next.js*, framework ini memiliki beberapa fitur utama yang menjadikannya pilihan populer di kalangan pengembang web:

1) *Server-Side Rendering* (SSR)

   SSR memungkinkan konten halaman web untuk dirender di server sebelum dikirim ke browser pengguna, menghasilkan HTML lengkap. Hal ini sangat meningkatkan kecepatan muat dan SEO karena konten dapat diakses langsung tanpa menunggu JavaScript dijalankan di klien.

2) *Static Site Generation* (SSG)

   SSG memungkinkan halaman web untuk dihasilkan secara statis selama proses build. Ini sangat cocok untuk konten yang tidak sering berubah, seperti blog atau dokumentasi, karena performa lebih baik dan halaman dapat disajikan langsung tanpa beban server.

3) A*PI Routes*

   Dengan *API Routes*, pengembang dapat membangun API langsung di dalam aplikasi Next.js. Ini memungkinkan Next.js berfungsi sebagai *framework full-stack* yang menangani *backend* dan *frontend* secara bersamaan.

4) *Integrasi TypeScript* dalam Next.js

TypeScript didukung sepenuhnya di Next.js, memberikan pengetikan statis untuk mendeteksi kesalahan pada waktu pengembangan. Ini meningkatkan kualitas dan keamanan kode serta mempermudah *refactoring*.

2.7 React Native

React Native adalah *framework* pengembangan aplikasi mobile berbasis *JavaScript* dan *React.js* yang memungkinkan pengembang membuat aplikasi native untuk platform *iOS* dan *Android*. Dikembangkan oleh Facebook pada tahun 2015, React Native dengan cepat menjadi salah satu kerangka kerja yang banyak digunakan dalam pengembangan aplikasi *mobile* karena mendukung penggunaan kembali kode (*code reusability*) dan efisiensi tinggi.

Menurut dokumentasi resmi React Native, beberapa fitur utama yang menjadikannya pilihan populer di kalangan pengembang adalah sebagai berikut:

1) Cross-Platform Development

Dengan React Native, sebagian besar kode aplikasi dapat ditulis sekali dan digunakan di kedua platform, iOS dan Android. Hal ini mengurangi waktu dan biaya pengembangan karena pengembang tidak perlu membuat dua aplikasi terpisah untuk masing-masing platform.

2) Komponen UI Asli (*Native UI Components*)

React Native memungkinkan penggunaan komponen UI asli (native UI components) di setiap platform. Dengan ini, aplikasi yang dibangun menggunakan React Native memiliki tampilan dan nuansa yang konsisten dengan standar antarmuka pengguna dari iOS dan Android, memastikan pengalaman pengguna yang.

3) Hot Reloading

Salah satu fitur unggulan React Native adalah hot reloading, yang memungkinkan pengembang melihat hasil perubahan kode secara instan tanpa harus memuat ulang aplikasi secara.

4) Integrasi dengan *Native Code*

React Native memudahkan integrasi dengan kode asli yang ditulis dalam bahasa seperti Java, Swift, atau Objective-C**.** Hal ini penting untuk fitur yang memerlukan performa tinggi atau mengakses fitur perangkat keras tertentu, seperti kamera atau sensor.

5) Dukungan Plugin Pihak Ketiga

React Native memiliki dukungan yang kuat untuk plugin pihak ketiga, yang memungkinkan pengembang untuk memperluas fungsionalitas aplikasi dengan mudah. Banyak plugin yang dibuat untuk menangani

tugas-tugas umum seperti geolokasi, integrasi kamera, hingga manajemen data.

2.8 *Cloud*

Cloud computing adalah sebuah proses pemanfaatan daya komputasi berbasis model *client-server* yang melibatkan penggunaan *CPU* dan *RAM.*, kecepatan jaringan, perangkat lunak, sistem operasi, maupun penyimpanan melalui jaringan internet. Cloud computing adalah model layanan infrastruktur yang memungkinkan pengguna untuk mengakses sumber daya komputasi secara jarak jauh melalui *internet* tanpa perlu memiliki atau memelihara perangkat keras secara fisik. Transfer data dilakukan secara non-fisik, dengan sumber daya komputasi yang terletak di lokasi penyedia layanan cloud.

Seiring bertambahnya jumlah pengguna komputer dan perangkat seluler, penyimpanan data kini menjadi fokus utama di hampir setiap bidang bisnis. Cloud computing memungkinkan perusahaan dari berbagai skala untuk memanfaatkan infrastruktur komputasi tanpa harus menanggung biaya tinggi untuk membangun infrastruktur TI internal. *Cloud services* seperti Infrastructure *as a Service (IaaS), Platform as a Service (PaaS)*, dan *Software as a Service (SaaS)* memungkinkan perusahaan mengembangkan dan mengelola aplikasi web dengan biaya yang lebih rendah dan fleksibilitas yang optimal (Alam., 2020).

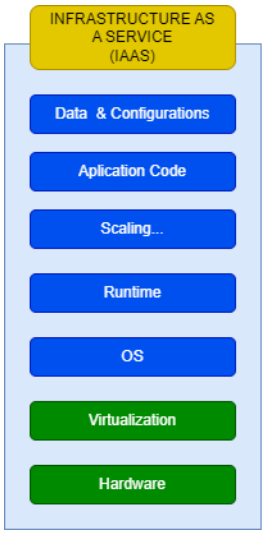
2.8.1 Mekanisme *Cloud Computing*

Cloud computing berfungsi sebagai media penyimpanan dan pengolahan data berbasis internet, yang dapat diakses secara fleksibel dimana saja dan kapan saja. Pengguna dapat mengakses server yang bekerja secara online di jaringan internet untuk menyimpan data dan memprosesnya. Layanan cloud memungkinkan penyimpanan dan pemrosesan data dilakukan di server jarak jauh, yang diakses melalui antarmuka seperti browser web. Keamanan dalam transmisi dan penyimpanan data menjadi perhatian utama dalam model layanan cloud, terutama dalam model layanan SaaS dan PaaS(Khoda Parast et al. 2022).

2.8.2 Klasifikasi *Cloud Computing*

Menurut (Google Cloud, 2024) *cloud computing* dapat dibagi menjadi beberapa klasifikasi. Berdasarkan lapisan layanan atau model infrastruktur yang di pakai. Setiap model memiliki peran dan manfaat tertentu dalam mendukung pengembangan aplikasi serta pengelolaan infrastruktur TI.

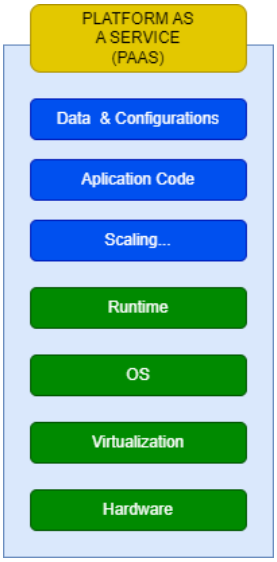
1) *Infrastructure-as-a-Service (IaaS)*



Gambar 6. *Infrastructure-as-a-service (IaaS)*

Layanan ini meliputi fasilitas seperti pusat data, penyimpanan, server, serta berbagai komponen jaringan yang dikelola oleh pihak ketiga. IaaS adalah salah satu model paling fleksibel, memfasilitasi pengguna dalam membangun dan mengelola infrastruktur di atas layanan yang disediakan oleh penyedia cloud

2) *Platform-as-a-Service (PaaS)*

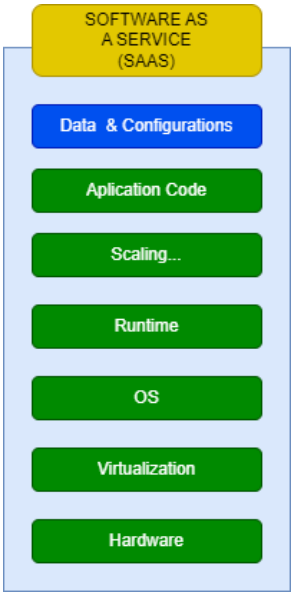


Gambar 7. *Platform-as-a-Service (PaaS)*

PaaS adalah layanan pengembangan platform berbasis web yang memungkinkan pengguna untuk membuat aplikasi web dengan cepat tanpa harus menangani pengelolaan infrastruktur di belakang layar.

3) *Software-as-a-Service (SaaS)*

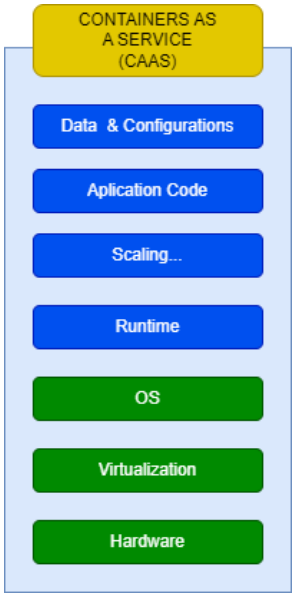
SaaS memungkinkan aplikasi berbasis web diakses oleh pengguna melalui internet tanpa perlu menginstal perangkat lunak di perangkat mereka. Model ini memungkinkan penggunaan perangkat lunak secara langsung dari penyedia layanan cloud.



Gambar 8. *Software-as-a-Service (SaaS)*

4) *Containers-as-a-Service (CaaS)*

Containers-as-a-Service (CaaS) adalah layanan yang menyediakan dan mengelola semua sumber daya perangkat keras dan perangkat lunak untuk mengembangkan dan menerapkan aplikasi menggunakan container. CaaS sering dianggap sebagai subkategori atau perpanjangan dari IaaS karena fokus utamanya adalah pada penggunaan container alih-alih mesin *virtual* (VM) sebagai sumber daya utama.



Gambar 9. *Containers-as-a-Service (CaaS)*

2.9 Firebase

Berdasarkan dokumentasi resmi Firebase. Firebase adalah platform yang membantu pengembang aplikasi dalam membuat dan menjalankan aplikasi lintas platform seperti Android, iOS, dan Web dengan lebih mudah. Firebase menawarkan *backend-as-a-service (BaaS)*, yang memungkinkan pengembang fokus pada pengalaman pengguna tanpa harus menangani infrastruktur backend.

Firebase menyediakan Realtime Database, layanan berbasis cloud yang menyimpan data dalam format JSON dan menyinkronkannya di berbagai perangkat secara real-time. Dengan dukungan untuk berbagai bahasa pemrograman, seperti Java, Node.js, dan lainya. Firebase memudahkan integrasi dengan platform yang berbeda. Layanan ini memungkinkan pengembang untuk fokus pada pengembangan aplikasi karena Firebase menangani banyak aspek teknis.

2.9.1 Perbandingan *Database Realtime dan Firestore Cloud*

Firebase menyediakan dua layanan database berbasis cloud yang dapat digunakan dalam berbagai aplikasi modern yaitu Database realtime dan Firestore Cloud. Meskipun keduanya adalah layanan NoSQL yang dihosting di cloud, masing-masing memiliki karakteristik yang berbeda dan dirancang untuk kasus penggunaan tertentu. Berdasarkan dokumentasi Firebase, berikut perbandingan dan keunggulan masing-masing:

Table 1. *Perbandingan Realtime Database dan Cloud firestore*

Aspek	Realtime Database	Cloud Firestore
Struktur Data	Menggunakan format JSON dengan satu pohon data besar	Menggunakan koleksi dan dokumen untuk struktur data yang lebih fleksibel.
Sinkronisasi *Real-Time*	Optimalkan untuk sinkronisasi real-time dengan perubahan langsung terlihat oleh pengguna.	Mendukung real-time dan lebih fokus pada penggunaan offline dengan sinkronisasi otomatis.
Offline Mode	Tidak mendukung mode offline.	Mendukung offline mode, memungkinkan aplikasi berjalan tanpa koneksi internet.
Skalabilitas	Cocok untuk aplikasi dengan data sederhana; kurang ideal untuk skala besar.	Lebih skalabel dengan dukungan kueri kompleks dan permintaan bersamaan.
Kinerja dan Query	Tidak mendukung kueri kompleks dengan baik	Mendukung kueri kompleks dengan indeks otomatis untuk kinerja optimal.
Kasus Penggunaan	Cocok untuk aplikasi chat dan notifikasi real-time.	Ideal untuk aplikasi e-commerce, CMS, dan aplikasi dengan data kompleks .

2.10 Metodologi Pengembangan Perangkat Lunak

Metodologi merupakan kerangka pijakan utama dalam perancangan dan pengembangan perangkat lunak profesional untuk menghasilkan sebuah sistem informasi yang sesuai dengan kebutuhan bisnis sebuah organisasi. Memilih sebuah metodologi bukanlah hal yang mudah dilakukan karena tidak satupun metodologi yang dapat dikatakan terbaik, metodologi pengembangan perangkat lunak diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada (Setiya Budi et al. 2016). Metodologi pengembangan perangkat lunak ini sangat diperlukan agar dapat lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak.

Komponen metodologi pengembangan perangkat lunak dapat dibagi ke dalam tiga unit, yaitu:

1. Metode, yaitu suatu cara atau teknik pendekatan yang sistematik yang dipergunakan untuk mengembangkan perangkat lunak. Metode ini mencakup: Perencanaan proyek dan perkiraan, analisis keperluan sistem dan perangkat lunak, perancangan struktur data, arsitektur program, prosedur *algoritma*, penulisan kode program (coding), uji coba, dan pemeliharaan.
2. Alat Bantu (Tools), yaitu alat-alat (manual maupun otomatis) yang mendukung pengembangan perangkat lunak. Terdapat dua alat bantu yang dapat digunakan yaitu: alat bantu manual dan alat bantu otomatis.
3. Prosedur, yang dipergunakan untuk mendefinisikan urut-urutan pekerjaan (daur) dari metode dan alat bantu tersebut (Pressman dan Maxim 2019).

Software Development Life Cycles (SDLC) adalah tahapan-tahapan pekerjaan yang dilakukan oleh analis sistem dan programmer dalam membangun sistem informasi (Dwanoko, 2016). Salah satu metode pengembangan sistem informasi yang popular pada saat sistem informasi pertama kali dibuat adalah metode System Development Life Cycle (SDLC) (Bolung et al. 2017).

Terdapat empat tahapan dalam membangun atau mengembangkan sistem informasi dengan menggunakan SDLC, yaitu: planning, analysis, design, dan implementation. Proses pengembangan perangkat lunak mengikuti tahap-tahap sebagai berikut:

a) Planning Sebuah perencanaan yang biasanya lebih menonjolkan pada alasan apa sebuah sistem dibuat dan dikerjakan oleh perangkat lunak dalam satu rentang waktu tertentu.
b) Analysis Tingkatan perencanaan yang dilanjutkan dengan proses analisis yang menekankan bagaimana, siapa, dan dimana system dibuat, mencakup arsitektur, antar muka internal, dan algoritma perangkat lunaknya
c) Design Proses desain lebih menekankan pada bagaimana system ini akan berjalan, penerapan dan pengujian unit-unit program.
d) Implementation Tahap ini mengenai proses penyampaiannya pada pengguna, integrasi dan pengujian modul-modul program. Setelah semua tahapan dilakukan maka perlu juga dikakukan pengujian sistem sebagai validasi perangkat lunak secara keseluruhan.

Software Development Life Cycles (SDLC) adalah proses keseluruhan dalam pembangunan sistem melalui beberapa proses atau tahapan. Beberapa

model pengembangan yang sering digunakan adalah waterfall, prototype dan RAD (Rapid Application Development (Bolung et al. 2017). Kelebihan dari model waterfall yaitu mudah dalam pengelolaan kebutuhan telah telah diindentifikasikan dan didokumentasikan, serta tahapan model waterfall berurutan secara linier. Sedangkan kelemahan dari model waterfall adalah tahapan yang berurutan secara linier tidak memungkinkan untuk kembali ke tahapan selanjutnya dan hampir tidak ada toleransi kesalahan, terutama pada tahapan planning dan design. Kelebihan dari model prototype yaitu proses identification yang akurat karena dilakukan evaluasi secara berkala dan mendapatkan masukan dari client terhadap prototype yang dihasilkan. Sedangkan kelemahan dari model prototype adalah setiap evaluasi dan masukan terhadap prototype, maka akan membutuhkan penyesuaian terhadap prototype tersebut serta terdapat kebutuhan biaya tambahan terkait dengan

pembuatan prototype dan dilakukan penyesuaian sesuai kebutuhan hingga prototype dapat disetujui oleh project owner. Kelebihan dari model RAD (Rapid Application Development) yaitu waktu siklus dapat pendek dengan penggunaan alat-alat RAD yang kuat serta produktivitas dengan lebih sedikit. Sedangkan kelemahan dari model RAD adalah cocok untuk sistem yang berbasis komponen dan terukur, serta membutuhkan personal yang sangat terampil (Bolung et al. 2017). Berikut ini merupakan perbandingan model Software Development Life Cycles (SDLC).

Table 2. Perbandingan Model Software Development Life Cycles (SDLC)

Model/ Features	**Waterfall**	**V-shape**	**CMM**	**RUP**	**Prototype**	**Increment al**	**Spiral**	**RAD**	**Agile**
Requireme nt specificatio n	*Beginning*	*Beginning*	*At second level*	*Beginning*	*Freq uently change d*	*Beginning*	*Beginning*	*Time- box released*	*Frequentl y changed*
Understandi ng requirements	*Well understoo d*	*Easily understoo d*	*Easily uderstoo d*	*Difficult to understan d*	*Not well understoo d*	*Well understood*	*Well understoo d*	*Well understoo d*	*Well understoo d*
Cost	*Low*	*Expensive*	*High*	*Expensive*	*High*	*Low*	*Expens ive*	*Low*	*Very High*
Guarantee of success	*Low*	*High*	*High*	*Not guaranteed*	*Good*	*High*	*High*	*Good*	*Very high*
Resourece Control	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*Yes*	*Yes*	*No*
Cost control	*Yes*	*Yes*	*Varies*	*Yes*	*No*	*No*	*Yes*	*Yes*	*Yes*
Simplicity	*Simple*	*Interme di ate*	*Interme di ate*	*Simple and clear*	*Simple*	*Simple*	*Interm e diate*	*Interm e diate*	*Complex*
Risk involveme nt	*High*	*Low*	*Varies acc to level*	*Critical risk in the early stages*	*low*	*Easily manag e*	*Low*	*Ver y Low*	*Reduced*
Expertise requ Ired	*High*	*Medium*	*Varies acc to level*	*Yes*	*Medium*	*High*	*High*	*Medi u m*	*Very High*
Change Incorporated	*Difficult*	*Difficult*	*Medium*	*Easy*	*Easy*	*Easy*	*Easy*	*Easy*	*Difficult*
Risk analysis	*Only at beginnin g*	*Yes*	*Yes*	*Yes*	*No risk analys is*	*No risk analys is*	*Yes*	*Low*	*Yes*
User involv ement	*Only atthe begining*	*Only atthe beginnig*	*Only atthe beginnig*	*At the beginning and the*	*High*	*Interm edi ate*	*High*	*Only at the beginni ng*	*High*

Model/ Features	**Waterfall**	**V-shape**	**CMM**	**RUP**	**Prototype**	**Incremental**	**Spiral**	**RAD**	**Agile**
				last phase					
Overlapping phases	*No such phase*	*No*	*No*	*Yes*	*Yes*	*No*	*Yes*	*No*	*Yes*
Flexibility	*Rigid*	*Little flexible*	*Highly flexible*	*considerable*	*Higly flexible*	*Less flexible*	*Flexible*	*High*	*Highly flexible*
Maintenace	*Least glamoro u s*	*Least*	*Typical*	*Promote maintainability*	*Routin e mainte na ce*	*promot e mainta ina bility*	*Typical*	*Easily maintai ned*	*Promote maintainab i l ity*
Integrity and security	*vital*	*Limited*	*Limited*	*Veryimporta nt*	*Weak*	*Robust*	*High*	*Vital*	*Demonstra b l e*
Reusability	*Limited*	*To some extend*	*Yes*	*Supports reusability of exiting classes*	*Weak*	*Yes*	*Yes*	*Some extend*	*Usecase reuse*
Interface	*Minimal*	*Minimal*	*Crucial*	*User interface*	*Crucial*	*Crucial*	*Crucial*	*Minimal*	*Model-driven*
Documentation & Training required	*Vital*	*Yes*	*Yes*	*Yes*	*Weak*	*Yes*	*Yes*	*Limited*	*Yes*
Time Frame	*Long*	*Acc to project size*	*Quite Long*	*Short time frame*	*Short*	*Very long*	*Long*	*Short*	*Least possible*

Sumber: (Ganpatrao Sabale, 2012)

2.11 Model Prototype

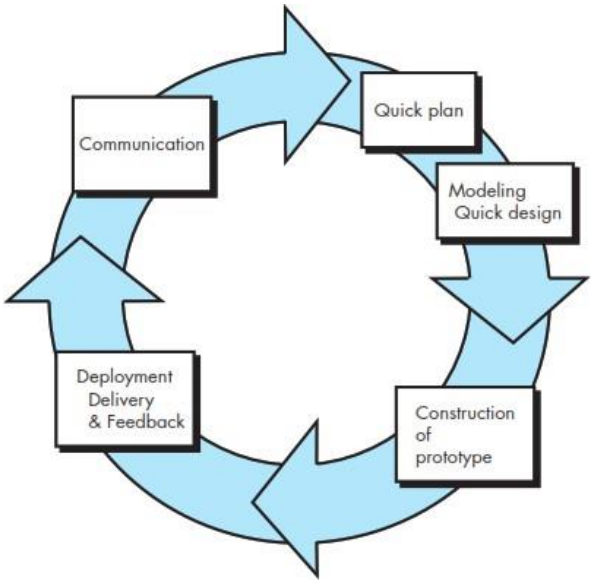
Prototype merupakan salah satu model dari SDLC. Model ini merupakan pengembangan dari Waterfall yang gabungan dengan model iterative. Prototype adalah sebuah versi awal dari perangkat lunak yang digunakan untuk mendemonstrasikan konsep, mencoba berbagai pilihan desain, dan menggali lebih banyak permasalahan dan solusinya. Dengan metode prototyping ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Prototyping, dimulai dengan pengumpulan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat, mendefinisikan objektif keseluruhan dari software, mengidentifikasikan segala kebutuhan, kemudian dilakukan perancangan kilat yang difokuskan pada penyajian aspek yang diperlukan agar pelanggan lebih terbayang dengan apa yang sebenarnya diinginkan. Kelebihan dari model prototype yaitu;

1. *Requirement identification* yang akurat karena dilakukan evaluasi secara berkala dan mendapatkan masukan dari client terhadap prototype yang dihasilkan
2. *User experience* yang meningkat, karena secara terus menerus melakukan uji coba dan evaluasi terhadap prototype
3. Kesalahan dan redundansi dapat diminimalkan karena proses identifikasi yang baik terhadap prototype.

Prototype akan dihilangkan atau ditambahkan pada bagiannya sehingga sesuai dengan perencanaan dan analisis yang dilakukan oleh pengembang sampai dengan uji coba dilakukan secara simultan seiring dengan proses pengembangan. Berikut merupakan tahapan-tahapan dalam Prototyping menurut (Pressman dan Maxim 2019):

1. *Communication*, yaitu tahap komunikasi dengan stakeholders untuk menentukan tujuan umum dari software, mengidentifikasi requirement yang sudah diketahui dan garis besar area yang sudah wajib ada.
2. *Quick plan*, yaitu tahapan perencanaan pembuatan sistem secara cepat segera setelah komunikasi terjadi.
3. *Modelling Quick Design*, yakni membuat model sistem serta membuat desain dari sistem secara cepat dengan berfokus pada aspek software yang terlihat oleh pengguna misalnya antarmuka sistem (GUI) atau format tampilan keluaran.
4. *Construction of Prototype*, desain yang telah dibuat lalu dibangun menjadi sebuah prototype.

5. *Deployment Delivery and Feedback*, yaitu tahapan dimana prototype yang sudah dibangun diperlihatkan dan dievaluasi oleh stakeholders untuk kemudian dapat memberikan umpan balik yang digunakan untuk mengidentifikasi requirement lebih lanjut dan sehingga bagi pengembang menjadi lebih memahami apa yang perlu dilakukan.



Gambar 10. Model Prototype (Pressman dan Maxim 2019)

Prototyping dapat diterapkan pada pengembangan sistem kecil maupun besar dengan harapan agar proses pengembangan dapat berjalan dengan baik, tertata serta dapat selesai tepat waktu. Keterlibatan aktif pengguna selama proses pembentukan prototipe akan memberikan manfaat besar bagi semua pihak terkait, termasuk pimpinan, pengguna itu sendiri, dan pengembang sistem. Manfaat lainnya dari penggunaan prototyping adalah:

1. Mewujudkan sistem sesungguhnya dalam sebuah replika sistem yang akan berjalan, menampung masukan dari pengguna untuk kesempurnaan sistem.
2. Pengguna akan lebih siap menerima setiap perubahan sistem yang berkembang sesuai dengan berjalannya prototype sampai dengan hasil akhir pengembangan yang akan berjalan nantinya.
3. Prototype dapat ditambah maupun dikurangi sesuai berjalannya proses pengembangan. Kemajuan tahap demi tahap dapat diikuti langsung oleh pengguna.
4. Penghematan sumber daya dan waktu dalam menghasilkan produk yang lebih baik dan tepat guna bagi pengguna.

Kunci agar model prototype ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang harus setuju bahwa prototype dibangun untuk mendefinisikan kebutuhan. Model prototype ini dapat membantu pengembang dan pengguna untuk membuat sistem yang sesuai dengan rencana, karena kebanyakan dari pengguna sering mengalami kesulitan dalam penyampaian kebutuhannya secara detail. Pengumpulan kebutuhan sebaiknya harus sudah disepakati terlebih dahulu antara pengembang dan pengguna agar proyek dapat berjalan sesuai waktu yang direncanakan.

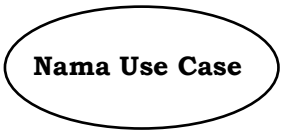
2.12 Unified Modeling Language (UML)

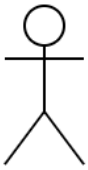
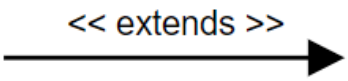
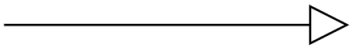
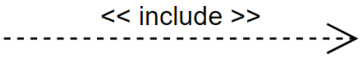
Rosa A. S dan Shalahuddin (2018:133) menyatakan bahwa Unified Modeling Language (UML) adalah salah satu bahasa standar yang banyak diterapkan di dunia industri untuk mendefinisikan kebutuhan, mengembangkan analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

a) Use Case Diagram

Use case atau diagram use case menurut Rosa A. S dan M. Shalahuddin tahun (2018:155) merupakan suatu pemodelan untuk kelakuan sistem informasi yang akan dibuat. Use case menggambarkan interaksi antara satu atau lebih aktor dengan sistem informasi yang sedang dirancang. Secara umum, use case digunakan untuk mengidentifikasi fungsi-fungsi dalam sistem informasi serta menentukan siapa saja yang memiliki hak akses terhadap fungsi-fungsi tersebut. Salah satu simbol yang digunakan dalam diagram use case dijelaskan pada tabel berikut ini

Table 3. Use Case Diagram

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor biasanya dinyatakan dengan menggunakan kata kerja di awal frasa nama use case.
Aktor/*actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang

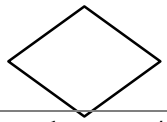
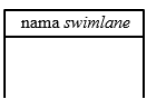
Simbol	Deskripsi
	akan dibuat berada di luar sistem informasi itu sendiri. Jadi, meskipun simbol aktor berupa gambar orang, aktor belum tentu manusia dan biasanya dinyatakan dengan kata benda di awal frasa nama aktor.
Asosiasi/*association*	Komunikasi berlangsung antara aktor dan use case, dan use case berinteraksi secara langsung dengan aktor.
Ekstensi/ extend 	Menunjukkan extension dari sebuah use case untuk menambahkan perilaku opsional. Arah panah bergerak dari extension use case menuju base use case.
Generalisasi/*generalization* 	Relasi generalisasi dan spesialisasi menggambarkan hubungan umum-khusus antara dua use case, di mana salah satu fungsi merupakan fungsi yang lebih umum dibandingkan dengan fungsi lainnya.
Menggunakan/*Include/uses* 	Menunjukkan inclusion fungsionalitas dari sebuah use case dengan use case lainnya. Arah panah ditujukan dari base use case ke included use case.

Sumber: Rossa A.S dan Shalahuddin (2018:156-158)

b) Activity Diagram

Menurut pendapat Rosa A.S dan M. Shalahuddin (2018:161) bahwa diagram aktivitas atau activity diagram adalah suatu workflow atau aliran kerja atau aktivitas yang ada dalam sistem atau proses bisnis atau dalam sebuah menu yang terdapat pada perangkat lunak. Poin penting yang perlu diperhatikan di sini adalah bahwa diagram aktivitas menggambarkan aktivitas yang dilakukan oleh sistem, bukan yang dihasilkan oleh aktor

Table 4. Activity Diagram

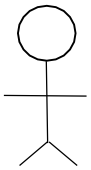
Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilaksanakan oleh sistem umumnya dimulai dengan kata kerja dalam pernyataannya.
Percabangan/ *decision* 	Percabangan digunakan untuk menghubungkan beberapa pilihan aktivitas ketika terdapat lebih dari satu alternatif yang dapat dijalankan.
Penggabungan/ *join* 	Penggabungan digunakan untuk menyatukan beberapa aktivitas yang berbeda menjadi satu alur aktivitas.
Status akhir 	Status akhir menunjukkan penyelesaian aktivitas sistem, di mana setiap diagram aktivitas memiliki titik akhir yang menandakan berakhirnya proses.
Swimlane  	Memisahkan organisasi bisnis bertujuan untuk menetapkan tanggung jawab atas setiap aktivitas yang terjadi, memastikan setiap bagian berfokus pada perannya masing-masing.

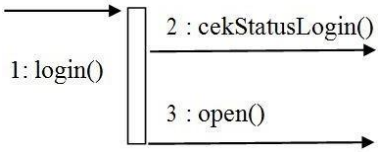
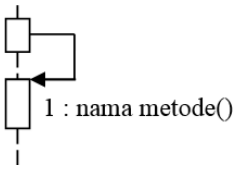
Sumber: Rosa A. S dan Shalahuddin (2018:162)

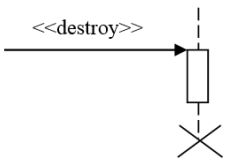
c) Diagram Sequence

Sequence Diagram menunjukkan bagaimana sistem merespons interaksi yang dilakukan oleh pengguna. Diagram ini secara langsung berhubungan dengan aktivitas utama dalam sistem informasi pengelolaan anggaran pendapatan dan belanja desa yang berbasis objek. Sementara itu, menurut Sukamto dan Shalahuddin (2018:165), "Diagram sequence menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek beserta pesan (message) yang dikirimkan dan diterima antar objek."

Table 5. Sequence Diagram

No.	Simbol	Deskripsi
1.	Aktor  Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda di awal frase nama aktor.
2.	Garis hidup/*lifeline* 	Menyatakan kehidupan suatu objek
3.	Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang di lakukan

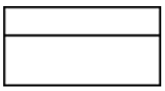
NO	Simbol	Deskripsi
		di dalamnya, misalnya  Maka cekStatusLogin() dan open() dilakuka didalam metode
5.	Pesan tipe *create* <<create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6.	Pesan tipe *call* 1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
7.	Pesan tipe *send* 1 : masukkan	Menyatakan pada objek yang dikirim data/masukan/ informasi ke objek lainnya arah panah mengarah pada objek dikirim. Bahwa suatu objek mengirimkan data/ masukkan /informasi ke objek lainnya, arah panah mengarah
8.	Pesan tipe *return* 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu,

No	Simbol	Deskripsi
9.	Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada *Destroy*

d) *Class Diagram*

Class diagram adalah representasi visual yang menunjukkan struktur sistem dari sisi perancangan kelas-kelas yang digunakan dalam pembangunan sistem. Diagram ini menampilkan struktur statis sistem, termasuk definisi atribut, metode (operasi), serta relasi atau keterkaitan antar kelas yang ada. Selain itu, Class diagram juga menggunakan simbol-simbol khusus untuk menggambarkan elemen-elemen tersebut secara sistematis.

Table 6. *Class Diagram*

No	Simbol	Nama	Keterangan
1.		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
2.		*Directed* Association	Merupakan asosiasi dengan makna kelas yang satu digunakan oleh kelas yang lain.
3.		Association	Menggambarkan kelas yang memiliki atribut berupa kela lain atau kelas yang harus engetahui eksistensi kelas lain
4.		*Agregation*	Relasi antarkelas dengan makna semua bagian (*whole-part*).
5.		*Generalization*	Relasi antarkelas dengan makna generalisasi-

No	Simbol	Nama	Keterangan
			spesialisasi (umum khusus).
6.		*Composition*	Merupakan bentuk khusus dari *agregation* dimana kelas yang menjadi bagian diciptakan setelah kelas *whole* dibuat

2.13 Pengujian Sistem

Dalam proses kegiatan pengembangan, perangkat lunak harus diuji untuk mengetahui apakah perangkat lunak telah sesuai keinginan karena proses dari analisis, perancangan serta pemrograman tidak bebas dari kesalahan. Pada proses pengujian diharapkan dapat menemukan kesalahan pada perangkat lunak yang telah dibangun agar mendapatkan perangkat lunak yang dengan kualitas yang sangat baik. Pengujian Perangkat lunak merupakan proses eksekusi suatu program atau sistem dengan maksud menemukan atau, melibatkan setiap kegiatan yang bertujuan untuk mengevaluasi atribut atau kemampuan suatu program atau sistem dan menentukan bahwa itu memenuhi hasil yang dibutuhkan perusahaan.

Pentingnya pengujian aplikasi sangat diperlukan dalam penilaian kualitas aplikasi yang ada, agar fungsi dalam aplikasi dapat bekerja sesuai dengan yang diharapkan pengguna (Nurudin et al. 2019). Pengujian perangkat lunak bertujuan utama untuk memastikan bahwa hasil akhir perangkat lunak telah sesuai dengan spesifikasi dan kebutuhan yang telah ditetapkan sebelumnya.

Black-Box Testing

Metode Black-box Testing merupakan pendekatan yang digunakan untuk menguji perangkat lunak tanpa memperhatikan detail struktur internalnya. Pendekatan ini menitikberatkan pada pengujian fungsionalitas aplikasi dengan memberikan input tertentu dan mengevaluasi apakah output yang dihasilkan sesuai dengan spesifikasi yang telah ditentukan. Proses pengujian ini sangat relevan untuk memastikan bahwa program berjalan sesuai dengan kebutuhan pengguna dan persyaratan yang telah ditetapkan. Misalnya, penelitian oleh

(Setiana et al. 2024) menunjukkan bahwa pengujian perangkat lunak dengan metode black-box pada aplikasi sistem pakar dapat membantu memastikan kebenaran pola operasi program dan validitas hasilnya. Selain itu, (Asfinoza et al. 2018) menegaskan bahwa pengujian dengan metode ini memberikan gambaran akurat terhadap kondisi dan fungsi aplikasi, memastikan bahwa sistem dapat memenuhi ekspektasi pengguna. Oleh karena itu, Black-box Testing menjadi salah satu metode yang efektif untuk menilai fungsionalitas perangkat lunak secara komprehensif tanpa harus memahami detail implementasi program.

User Accepted Testing

User Acceptance Testing (UAT) adalah tahap akhir dalam siklus pengujian perangkat lunak yang dilakukan oleh pengguna akhir untuk memastikan bahwa sistem memenuhi kebutuhan dan tujuan bisnis yang telah ditetapkan. Menurut (Hambling et al. 2013) UAT tidak hanya mengevaluasi sistem dari sisi teknis tetapi juga dari segi kemampuannya mendukung proses bisnis nyata yang dihadapi oleh pengguna sehari-hari. Pengujian ini melibatkan skenario yang realistis dengan data operasional untuk menilai apakah sistem dapat berjalan sesuai harapan dalam kondisi sebenarnya.

Keberadaan UAT menjadi sangat penting karena berfungsi sebagai tahap terakhir sebelum sistem digunakan secara penuh dalam organisasi. Dengan melibatkan pengguna langsung dalam pengujian, potensi masalah yang mungkin terlewatkan pada tahap pengembangan dapat diidentifikasi lebih dini. Selain itu, UAT memberikan jaminan kepada pemangku kepentingan bahwa sistem telah siap diimplementasikan. Penelitian terbaru oleh (Wang et al. 2024) menunjukkan bahwa teknologi berbasis kecerdasan buatan seperti *XUAT-Copilot* dapat mengotomatisasi banyak aspek UAT, memungkinkan pengujian dilakukan dengan lebih efisien dan dalam skala besar. Pendekatan ini juga mengurangi risiko kesalahan manual yang sering kali menjadi tantangan dalam pengujian tradisional.

Proses UAT, sebagaimana dijelaskan oleh Hambling dan van Goethem, terdiri dari lima langkah utama. Tahap pertama adalah perencanaan, yang mencakup penentuan tujuan pengujian, kriteria penerimaan, dan cakupan pengujian. Selanjutnya adalah desain tes, di mana skenario pengujian dikembangkan berdasarkan kebutuhan bisnis dan alur kerja pengguna. Pelaksanaan pengujian dilakukan oleh pengguna akhir dengan menguji setiap skenario yang telah dirancang. Setelah itu, hasil pengujian dievaluasi untuk melihat apakah sistem memenuhi kriteria penerimaan yang telah ditetapkan.

Terakhir, keputusan diambil apakah sistem layak untuk diimplementasikan atau perlu perbaikan lebih lanjut. Proses ini memastikan bahwa setiap aspek sistem telah diuji secara mendalam sebelum digunakan secara resmi.

Pengujian UAT terdiri atas 2 pengujian yaitu Alpha Testing dan Beta Testing (Maindra et al. 2023), yang keduanya berperan penting dalam memastikan kesiapan sistem sebelum peluncuran resmi. Alpha Testing dilakukan di lingkungan internal oleh tim pengembang atau QA. Tujuannya adalah untuk mendeteksi bug, kekurangan fungsi, dan ketidaksesuaian terhadap kebutuhan bisnis sebelum sistem diperkenalkan ke pihak luar. Sementara itu, Beta Testing melibatkan pengguna eksternal atau pelanggan sesungguhnya, yang menguji sistem dalam kondisi nyata. Umpan balik dari beta tester sangat berharga untuk mengidentifikasi masalah yang mungkin tidak terdeteksi selama pengujian internal, termasuk aspek performa, kegunaan, dan kompatibilitas. Kombinasi kedua bentuk UAT ini memberikan pendekatan bertahap dalam validasi akhir sistem dimulai dari lingkungan internal, hingga lingkungan eksternal yang mencerminkan situasi dunia nyata.

2.14 Perbandingan Penelitian Terdahulu

Dalam studi ini, penulis merujuk pada beberapa hasil penelitian terdahulu yang relevan sebagai acuan dan bahan pertimbangan untuk memperkaya analisis yang dilakukan. Hasil penelitian terdahulu ini diharapkan dapat memberikan wawasan tambahan serta memperkuat landasan teori yang digunakan. Rincian lengkapnya disajikan pada Tabel Penelitian Terdahulu, yang memuat berbagai temuan relevan dari penelitian-penelitian sebelumnya.

Table 7. Penelitian Terdahulu

No	Judul Penelitian	Peneliti	Tujuan	Hasil Penelitian
1	Sistem Informasi Presensi Karyawan Berbasis Android (Studi Kasus: Asuransi Panin Dai-Ichi Life)	(Ramadhini et al. 2023)	Mengembangkan aplikasi presensi berbasis Android dengan teknologi GPS untuk mencatat dan mempercepat proses presensi.	Sistem berhasil mempercepat proses presensi dengan memanfaatkan GPS, serta mencatat lokasi karyawan secara akurat.

No	Judul Penelitian	Peneliti	Tujuan	Hasil Penelitian
2	Implementasi Face Recognition dan Geolocation Pada Sistem Presensi Karyawan Berbasis Mobile Apps	(Wibowo et al. 2024)	Mengimplementasikan teknologi face recognition dan geolocation pada sistem presensi karyawan berbasis mobile Android.	Sistem berhasil mempermudah pencatatan dan verifikasi data presensi, dengan pendeteksian wajah yang akurat dan lokasi presensi melalui GPS.
3	Presensi Karyawan Perguruan Tinggi Berbasis Aplikasi Mobile Menggunakan Geolocation dan Verifikasi Biometrik	(Ardianto et al. 2022)	Membuat sistem presensi karyawan berbasis aplikasi mobile menggunakan teknologi geolocation dan validasi biometrik untuk monitoring presensi secara real-time.	Sistem presensi berhasil mencatat kehadiran dengan akurasi 90,9% dalam pengujian geolocation dan biometric, serta mengurangi risiko kecurangan.
4	Sistem Informasi Manajemen Presensi Siswa Berbasis Mobile (Studi Kasus: SMA N 1 Sungkai Utara)	(Putra Setiawan 2021)	Mengembangkan sistem presensi berbasis mobile untuk mempermudah proses presensi siswa serta pemberitahuan ke orang tua.	Sistem berhasil mempercepat dan mempermudah proses presensi siswa, serta memberikan notifikasi otomatis kepada orang tua.
5	Pendekatan Metode Prototype Pada Aplikasi Presensi Berbasis	(Pratama et al. 2023)	Mendesain aplikasi presensi berbasis mobile untuk meningkatkan efisiensi pengelolaan data kehadiran di	Aplikasi berhasil meningkatkan akurasi pengelolaan data kehadiran dan meminimalkan

No	Judul Penelitian	Peneliti	Tujuan	Hasil Penelitian
	Mobile (Studi Kasus: Kantor Desa Mekar Jaya)		kantor desa dengan metode prototype.	kesalahan dalam pencatatan presensi.
6	Aplikasi Presensi Berbasis Mobile Menggunakan Teknologi Geolocation dan Face Recognition	(Abel et al. 2024)	Mengembangkan aplikasi presensi yang menggabungkan teknologi geolocation dan face recognition untuk mengatasi kecurangan dalam proses presensi online yang dilakukan oleh pegawai perusahaan swasta.	Aplikasi yang dikembangkan berhasil memastikan lokasi dan identitas pegawai saat melakukan presensi, sehingga meningkatkan akurasi dan keandalan data kehadiran.
7	Implementasi Location Based Service Pada Sistem Informasi Kehadiran Pegawai Berbasis Android	(Qois et al. 2021)	Mengembangkan sistem kehadiran pegawai berbasis aplikasi mobile yang menggunakan teknologi GPS dan validasi selfie untuk memverifikasi kehadiran secara real-time.	Sistem memungkinkan pelaporan presensi secara real-time dengan validasi lokasi dan gambar, serta mempermudah HRD untuk memantau lokasi pegawai selama WFH