BAB II
KAJIAN PUSTAKA

A. Kajian Teoritis

1. Aplikasi *mobile*

Menurut Habibi et al. (2020:7), aplikasi adalah suatu program komputer yang bertujuan untuk menjalankan tugas dari user. Sedangkan menurut Tolle et al. (2017:13), aplikasi adalah perangkat lunak yang berjalan pada *smartphone*, tablet atau perangkat yang sejenis. Menurut Adil et al. (2023:233), aplikasi adalah perangkat lunak yang digunakan untuk menjalankan sesuatu pada sistem komputer dan tidak hanya digunakan untuk komputer namun aplikasi dapat digunakan di telepom seluler.

Mobile identik sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ketempat lain tanpa terjadi hambatan komunikasi (Rahmadana, 2021:49). Sedangkan menurut Prabowo et al. (2021:1), *mobile* diartikan sebagai perpindahan dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon dapat berpindah ke tempat lain dengan mudah tanpa terjadi pemutusan komunikasi .

Menurut Kurosaki (2020:2), *flutter* adalah kerangka kerja yang digunakan untuk mengembangkan aplikasi *Android*, tetapi tidak hanya itu, juga untuk perangkat iOS, aplikasi *web*, dan bahkan aplikasi desktop untuk *Windows*, *Linux*, dan *macOS*. *Flutter*, diciptakan oleh *google*, adalah sebuah

platform pengembangan perangkat lunak yang memungkinkan pengembang untuk membuat aplikasi yang kompatibel dengan berbagai sistem operasi seperti *Android*, iOS, *web, Linux*, dan *macOS* hanya dengan menggunakan satu kode sumber (Ismail et al., 2023:28)

Berdasarkan uraian tentang aplikasi *mobile* diatas dapat disimpulkan aplikasi *mobile* dapat diartikan sebagai sebuah perangkat lunak yang dirancang khusus untuk digunakan pada perangkat *mobile* seperti ponsel dan tablet. Aplikasi ini memungkinkan pengguna untuk melakukan berbagai aktivitas seperti mengakses informasi, berkomunikasi, bermain game, melakukan transaksi, dan masih banyak lagi.

Menurut Putra et al. (2023:30) jenis aplikasi *mobile* sebagai berikut:

a. *Native* apps (aplikasi asli)

   Pengembangan aplikasi asli melibatkan penulisan kode dalam bahasa yang didukung secara langsung oleh sistem operasi perangkat *mobile* tertentu. Misalnya, aplikasi asli untuk iOS dan Android. Pendekatan ini digunakan ketika pengembang ingin membuat aplikasi yang khusus ditujukan untuk *Apple App Store* atau *Google Play Store*. Metode ini sangat cocok untuk aplikasi yang membutuhkan penyesuaian tingkat tinggi yang memanfaatkan komponen asli masing-masing perangkat. Ini sangat ideal untuk aplikasi game, aplikasi VR, dan aplikasi yang memerlukan grafik yang intensif. Namun, satu set kode tidak dapat dijalankan di kedua platform ketika menggunakan pengembangan asli.

b. *Hybrid* apps (aplikasi hidup)

Aplikasi hibrid digunakan untuk mengembangkan aplikasi *mobile* yang dapat berfungsi di berbagai platform. Kode hanya ditulis sekali dalam satu bahasa dan dapat dijalankan di iOS maupun *Android*. Metode ini mempercepat proses pengembangan aplikasi *mobile* karena pengembang tidak perlu menulis kode dua kali untuk masing-masing sistem operasi. Meski begitu, dibandingkan dengan aplikasi *native*, pengembang akan kehilangan sedikit fleksibilitas dalam hal kemampuan aplikasi hibrid. Namun, pendekatan ini tetap menguntungkan bagi sebagian besar pengembang.

c. *Progressive web* app (WPA)

PWA adalah aplikasi ringan yang beroperasi melalui URL di browser *web* pada perangkat. Meskipun tampilannya menyerupai aplikasi *mobile*, aplikasi ini tidak di instal langsung pada perangkat. Pengembang web yang berpengalaman dapat dengan mudah membuat PWA karena menggunakan bahasa pemrograman yang sudah mereka kuasai, sehingga tidak memerlukan banyak waktu untuk belajar hal baru.

Menurut Hodijah (2024:2) karakteristik aplikasi *mobile* berdasarkan kebutuhan pengguna, yaitu :

a. Fitur aplikasi

Fitur-fitur yang dimiliki aplikasi *mobile* umumnya lebih spesifik, instan, dan terbatas, seperti SMS, membaca notifikasi, mengambil foto, dan mengunggahnya ke internet. Di mana pun pengguna

smartphone menggunakan aplikasi ini, mereka akan memperoleh produktivitas maksimum dengan usaha yang minimal.

b. Interaksi *user*

Dapat diakses dengan cara-cara yang sederhana, seperti melalui layar sentuh. Dengan fitur layar sentuh, pengguna *smartphone* dapat berinteraksi langsung dengan layar, serta menggunakan sensor *accelerometer* dan kompas.

c. Pemetaan lokasi

Dengan menggunakan GPS yang terintegrasi dalam *smartphone*, lokasi dapat dipetakan secara detail.

d. *Push notification*

Penggunaan teknologi *mobile* biasanya berlangsung dalam waktu singkat, seperti saat membaca notifikasi. Notifikasi ini dikirimkan kepada pengguna *smartphone* tanpa perlu login terlebih dahulu, dan pengguna hampir selalu berada dekat dengan *smartphone* mereka.

Menurut Ismail et al. (2023:29) beberapa dari kelebihan *flutter* adalah :

a. *Flutter* bersifat *open-source*

Pengembang memiliki kesempatan untuk mengikuti kemajuan *flutter*, berpartisipasi dalam penyelesaian *bug* atau isu yang ada, dan bahkan mencoba fitur-fitur baru yang sedang dalam tahap pengembangan. Selain itu, mereka juga dapat membuat fitur-fitur baru yang dapat digunakan oleh pengembang lainnya.

b. *Flutter* memiliki fitur *hot-reload*

Salah satu keunggulan utama *flutter* adalah *hot-reload*, yang memungkinkan pengembang untuk memperbarui tampilan aplikasi di emulator tanpa perlu *me-reload* halaman atau merangkai ulang kode. Selain itu, *flutter* juga memiliki fitur lain yang disebut *hot-restart*, yang berguna jika pengembang melakukan banyak perubahan pada struktur *widget* aplikasi atau ingin mengubah status aplikasi. *Hot-restart* membutuhkan sedikit lebih lama daripada *hot-reload* karena melibatkan pembaruan, *restart* aplikasi, dan reset status, tetapi lebih cepat daripada merangkai ulang aplikasi.

c. *Flutter* memiliki *toolkit* tampilan yang indah

Alat-alat pengembangan lintas platform lainnya seringkali memiliki antarmuka yang monoton. *Flutter*, sebaliknya, dirancang dengan menggunakan desain material dari *google* yang menarik dan siap pakai, serta menyediakan kemudahan bagi pengembang untuk menerapkan desain material *cupertino* untuk membuat antarmuka pada iOS. Tampilan aplikasi dapat disesuaikan untuk berbagai platform sehingga kesan produk akhirnya tetap konsisten dengan sistem operasi yang digunakan.

d. *Flutter* mendukung akses *libraries*

Flutter mendukung keterhubungan antarbahasa dengan bahasa pemrograman C dan C++, dan menyediakan saluran platform yang

efektif untuk terhubung dengan *Kotlin* atau *Java* untuk *Android*, serta *Swift* atau *Objective-C* untuk iOS.

e. *Flutter* memiliki UI yang fleksibel

Flutter menawarkan animasi dan peralihan yang estetis, dan pengembang memiliki kemampuan untuk membuat *widget* kustom yang memungkinkan kreativitas dalam desain antarmuka pengguna. Ini berarti *widget* dapat diatur dengan bebas, memungkinkan pengembang untuk berkreasi dengan UI sesuai keinginan mereka. Sebagai contoh, pengembang dapat menempatkan video di belakang *scrollview* dan menempatkan *toolbar* di atas *canvas*.

2. Presensi berbasis *face recognition*

Menurut Bastian (2007:117), presensi adalah alat dokumentasi jam kerja karyawan (termasuk lembur). Presensi juga dapat dirancang untuk menggabungkan informasi libur, cuti sakit, dan libur hari raya. Sedangkan menurut Mulyadi (2023:230), presensi adalah alat bukti *administrative* yang selalu dipantau pelaksanaanya oleh lembaga karena berkaitan dengan perhitungan pendapatan atau gaji para pegawai. Presensi juda dapat diartikan eksistensi atau keberadaan seseorang atau sekumpulam orang pada suatu tempat.

Menurut Suryansah et al. (2020:35), pengenalan wajah adalah teknologi komputer yang memungkinkan identifikasi atau verifikasi seseorang melalui gambar digital. Teknologi ini bekerja dengan

mencocokkan tekstur dan kontur wajah dengan data wajah yang tersimpan dalam basis data. Sedangkan menurut Adjabi et al. (2020:1), pengenalan wajah adalah salah satu bidang penelitian paling aktif dalam visi komputer dan pengenalan pola, dengan banyak aplikasi praktis dan komersial termasuk identifikasi, kontrol akses, forensik, dan interaksi manusia-komputer.

Berdasarkan uraian tentang presensi dan *face recognition* diatas maka dapat disimpulkan presensi menggunakan *face recognition* adalah sistem yang memungkinkan pendokumentasian jam kerja karyawan, termasuk lembur, dengan cara yang efisien dan akurat. Teknologi ini tidak hanya mencatat kehadiran karyawan, tetapi juga dapat mengintegrasikan informasi tentang libur, cuti sakit, dan libur hari raya. Selain itu, penggunaan pengenalan wajah sebagai alat presensi meningkatkan keamanan dan keandalan proses identifikasi dan verifikasi karyawan, mendukung administrasi yang lebih baik, serta memastikan ketepatan dalam perhitungan pendapatan atau gaji pegawai.

Menurut Suryansah et al. (2020:36), berdasarkan bidang-bidang dalam penelitian yang berkaitan dengan pemrosesan wajah (*face recognition*) antara lain adalah :

a. Pengenalan wajah (*face recognition*) adalah proses membandingkan gambar wajah yang dimasukkan dengan database wajah, dan menemukan data wajah yang paling sesuai dengan gambar tersebut.

b. Autentikasi wajah (*face authentication*) adalah proses memeriksa keabsahan atau kesesuaian sebuah wajah dengan data wajah yang telah dimasukkan sebelumnya.
c. Lokalisasi wajah (*face localization*) adalah proses deteksi wajah dengan asumsi bahwa hanya terdapat satu wajah dalam gambar.
d. Penjejakan wajah (*face tracking*) adalah proses mengestimasi posisi sebuah wajah dalam video secara langsung. Pengenalan ekspresi wajah adalah untuk mengidentifikasi emosi manusia dengan melihat ekspresi wajah.

Menurut Suryansah et al. (2020:37), cara kerja pada *face recognition* sistem yaitu :

a. Pendeteksian wajah melibatkan pengambilan gambar wajah manusia melalui pemindaian foto 2D secara digital atau menggunakan video untuk mendapatkan gambar wajah 3D.
b. Penjajaran melibatkan kemampuan perangkat lunak untuk menentukan posisi, ukuran, dan orientasi kepala setelah berhasil mendeteksi wajah. Perangkat lunak 3D dapat mengenali gambar wajah dalam rentang hingga 90 derajat, sedangkan untuk perangkat lunak 2D, posisi kepala harus menghadap kamera setidaknya dalam rentang 35 derajat.
c. Pengukuran dilakukan oleh perangkat lunak untuk mengevaluasi lekukan wajah dengan menggunakan skala sub-milimeter (*microwave*) dan membuat sebuah templat.

d. Representasi. Setelah template dibuat, ia dapat diubah menjadi kode unik yang mewakili setiap wajah.
e. Pencocokan. Jika representasi foto wajah dan data wajah dalam basis data keduanya dalam format 3D, proses pencocokan bisa dilakukan secara langsung. Namun, tantangan muncul ketika mencocokkan representasi 3D dengan basis data foto 2D. Teknologi terbaru saat ini sedang mengatasi masalah ini. Saat foto wajah 3D diambil, perangkat lunak akan mengenali beberapa titik penting, seperti mata luar dan dalam, serta ujung hidung (biasanya tiga titik). Dengan menggunakan hasil pengukuran ini, perangkat lunak akan mengonversi gambar 3D menjadi 2D, dan membandingkannya dengan gambar wajah 2D yang ada dalam basis data.
f. Verifikasi atau identifikasi. Verifikasi adalah langkah pencocokan satu-per-satu, sementara identifikasi melibatkan perbandingan foto wajah yang diambil dengan seluruh gambar yang memiliki kesamaan dalam basis data.
g. Analisis tekstur wajah telah mengalami perkembangan dalam perangkat lunak pengenalan wajah dengan memanfaatkan biometrik kulit atau karakteristik unik dari tekstur kulit untuk meningkatkan ketepatan pencocokan. Namun, beberapa faktor menghambat efektivitas analisis tekstur ini, seperti pantulan cahaya dari kacamata atau foto wajah yang mengenakan kacamata hitam. Hambatan lain termasuk rambut panjang yang menutupi bagian tengah wajah, pencahayaan yang tidak optimal

yang menyebabkan gambar wajah kelebihan atau kekurangan cahaya, serta resolusi yang rendah ketika foto diambil dari jarak yang jauh.

3. *Dart*

Menurut Wali et al. (2023:178), *dart* merupakan sebuah bahasa pemrograman yang telah dikembangkan oleh *google* sejak tahun 2012 di bawah kepemimpinan Lars Bak dan Kasper Lund, dan menjadi tersedia untuk digunakan oleh publik pada tahun 2013. Sedangkan menurut Kurosaki (2020:2), *dart* adalah bahasa pemrograman yang menjadi pilihan utama untuk mengembangkan kerangka kerja *flutter,* dan dikembangkan secara langsung oleh *google*.

Berdasarkan uraian tentang *dart* diatas dapat disimpulkan *dart* adalah bahasa pemrograman yang dikembangkan oleh *google* dan pertama kali dirilis pada tahun 2013. *Dart* dirancang untuk digunakan dengan kerangka kerja *flutter* untuk pengembangan aplikasi *mobile* dan *web*.

Menurut Wali et al. (2023:179), dart memiliki beberapa kegunaan, sebagai berikut :

a. *Dart* digunakan dalam konteks *flutter framework,* yang dibuat secara khusus untuk pembangunan antarmuka (UI), serta diterapkan dalam berbagai jenis aplikasi seperti aplikasi *web*, layanan mikro, aplikasi desktop, dan aplikasi seluler untuk platform iOS dan *Android*.
b. *Dart* diciptakan untuk menyederhanakan proses pengkodean dalam tugas pemrograman. *Dart* adalah bahasa pemrograman yang kuat dan

handal, sehingga cocok digunakan untuk mengembangkan produk aplikasi, baik dalam skala kecil maupun besar.

c. *Dart* juga dikompilasi secara langsung (*just-in-time*/JIT) untuk meningkatkan kecepatan dalam menampilkan perubahan kode program saat menggunakan fitur *hot-reload* dari *flutter* saat *men-debug* aplikasi di simulator atau emulator.

d. *Dart* didukung oleh beragam perpustakaan dan alat (*tools*) yang memungkinkan pembuatan aplikasi dalam skala besar dan dengan kecepatan yang tinggi.

e. *Dart* dan *flutter* merupakan kepemilikan *google*, sehingga pengendalian dan integrasi di antara keduanya menjadi lebih mudah. *Dart* berkembang seiring dengan *flutter*, dan seiring waktu, keduanya akan saling mendukung untuk meningkatkan kinerja dalam pengembangan perangkat lunak.

Menurut Wali et al. (2023:181), beberapa *library dart* yang umum digunakan dapat dilihat tabel dibawah ini :

Tabel 2. 1 Library Dart

Libary	Kegunaan
dart : core	Secara otomatis digunakan untuk semua operasi pokok dalam *dart*, termasuk pengelolaan *string*, koleksi, tanggal, dan URI.
dart : html	Digunakan untuk mengelola Dokumen Objek Model (DOM) adalah antarmuka pemrograman aplikasi (API) yang berfungsi untuk mengoperasikan dokumen HTML dan XML.
dart : io	Dukungan tersedia untuk berkas, soket, dan aplikasi non-*web* lainnya.

dart : async	Dukungan untuk pemrograman secara asinkron pada kelas *future/streams*.
dart : math	Digunakan untuk operasi matematika dan pembangkitan bilangan acak.
dart : developer	Digunakan untuk *debugging* atau inspeksi kode program.

Sumber: Wali et al. (2023), Pengantar 15 Bahasa Pemrograman Terbaik di Masa Depan (Referensi & Coding Untuk Pemula), hal. 181

4. *Firebase*

Menurut Wijaya et al. (2023:9), *firebase* adalah layanan *cloud* yang sering dimanfaatkan untuk mengembangkan aplikasi yang membutuhkan pertukaran data *real-time*, baik pada platform situs *web* maupun *mobile*. Integrasi dengan berbagai *framework* memungkinkan pertukaran data melalui API dalam format JSON. Proses sinkronisasi dilakukan dengan menggunakan basis data NoSQL.

Menurut Vikas et al. (2023:3), kelebihan dan kekurangan *firebase*, sebagai berikut :

a. Kelebihan :
    a. Memberikan notifikasi kepada pengguna secara langsung ketika terjadi perubahan data.
    b. Penerapan yang simple.
    c. Membolehkan pengembang untuk merespons perubahan data dengan cepat.
b. Kekurangan :
    a. Keterbatasan kontrol atas detail-detail modifikasi data.
    b. Perubahan versi yang sering dapat berdampak pada kinerja.

c. Pertimbangan dan penanganan dengan cermat terhadap kondisi persaingan dan kecepatan sangat penting.

5. *Flowchart*

Menurut Rahmi et al. (2022:81), *flowchart* merupakan representasi visual dari langkah-langkah dan urutan prosedur dari sebuah program. Ini membantu analis dan programmer dalam memecahkan masalah menjadi bagian-bagian yang lebih kecil serta membantu dalam mengevaluasi alternatif-alternatif dalam operasi. *Flowchar*t seringkali mempermudah pemecahan masalah, terutama ketika diperlukan pemahaman dan evaluasi lebih lanjut.

Menurut Rifka (2017:168), *flowchart* dalam *Standar Operasional Prosedur* (SOP) adalah diagram yang menggunakan simbol-simbol khusus untuk mengilustrasikan urutan langkah-langkah secara rinci dan hubungan antara satu proses atau instruksi dengan proses lain dalam program tersebut. *Flowchart* terdiri dari lima jenis yaitu :

a. *Flowchart* sistem

   Flowchart sistem adalah diagram yang menggambarkan alur kerja di dalam suatu sistem secara menyeluruh. Bagan ini menjelaskan urutan prosedur dan proses yang terhubung untuk membentuk sistem yang berfungsi.

b. *Flowchart* dokumen

   Flowchart dokumen, yang juga dikenal sebagai *form flowchart* atau *paper-work flowchart*, adalah diagram alur yang menggambarkan

perjalanan dari dokumen-dokumen, termasuk laporan dan formulir beserta salinannya. *Flowchart* dokumen menggunakan simbol-simbol yang serupa dengan yang digunakan dalam *flowchart* sistem.

c. *Flowchart* skematik

Flowchart skematik adalah bagan alir yang mirip dengan *flowchart sistem*, namun dengan fokus pada visualisasi prosedur di dalam sistem.

d. *Flowchart* program

Flowchart program adalah diagram yang menggambarkan dengan rinci langkah-langkah dari proses program. *Flowchart* program dibuat dengan mendasarkan pada *flowchart* sistem. Terdapat dua jenis *flowchart* program. Pertama, adalah diagram alur logika program yang digunakan untuk mewakili setiap langkah secara logis dalam program komputer, biasanya disiapkan oleh analis sistem. Dan kedua, adalah diagram alur program komputer yang sangat detail.

e. *Flowchart* proses

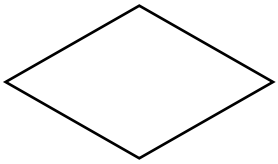
Flowchart proses adalah metode visual dalam rekayasa industri yang menguraikan dan menganalisis langkah-langkah berikutnya dalam suatu prosedur atau sistem. *Flowchart* proses menggunakan lima simbol khusus. Ini digunakan oleh insinyur industri untuk mempelajari dan meningkatkan proses-produksi. Dalam analisis sistem, *flowchart* proses sangat berguna untuk melacak jalur dari laporan atau formulir.

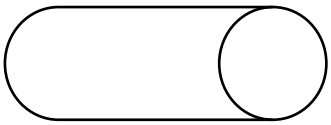
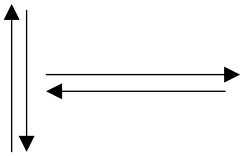
Menurut Rahmi et al. (2022:92), keuntungan penggunaan *flowchart* yaitu :

a. Sebagai dokumentasi prosedur kerja dalam ISO
b. Sebagai pedoman untuk menjalankan operasional
c. Sebagai pedoman untuk melakukan pelatihan terhadap karyawan baru
d. Sebagai *benchmark* (patokan)
e. Sebagai peta kerja untuk mencegah terjadi kehilangan arah
f. Untuk mempermudah pengambilan Keputusan

Menurut Rifka (2017:170), simbol-simbol yang digunakan dalam *flowchart* dapat dilihat pada tabel dibawah ini:

Tabel 2. 2 Simbol *Flowchart*

Simbol	**Nama**	**Deskripsi**
	Terminal	Simbol tersebut digunakan sebagai simbol untuk memulai maupun mengakhiri suatu program.
	Input/Output	Simbol ini digunakan sebagai simbol yang menyatakan sebuah input atau output tanpa melihat jenisnya.
	*Manual Operation*	Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh komputer.
	*Decision*	Simbol yang digunakan sebagai simbol untuk memilih sebuah proses dimana akan dilakukan berdasarkan suatu kondisi tertentu.

	*Processing*	Simbol yang digunakan sebagai simbol untuk menunjukkan pengolahan data dimana pengolahan data tersebut dilakukan oleh komputer.
	*Disk Storage*	Simbol yang digunakan sebagai simbol untuk menyatakan suatu masukan dan keluaran yang asalnya dari disk.
	*Flow Direction* Simbol atau Garis Koneksi	Simbol yang digunakan sebagai simbol untuk penghubung antar simbol lain dan menyatakan suatu arus proses.

Sumber : Rifka (2017), Step By Step Lancar Membuat SOP , hal.170

6. UML (*Unified Modeling Language*)

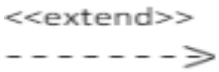
Menurut Destriana et al. (2021:1), *Unified Modeling Language* (UML) adalah suatu bahasa yang digunakan untuk menetapkan, menggambarkan, membangun, dan mendokumentasikan elemen-elemen (komponen-komponen dari informasi yang digunakan untuk menghasilkan proses pembuatan perangkat lunak, seperti model, deskripsi, atau perangkat lunak) dari sistem perangkat lunak, termasuk dalam pemodelan bisnis serta sistem non-perangkat lunak lainnya.

a. *Use case diagram*

Menurut Destriana et al. (2021:7), *use case* adalah alat untuk menggambarkan kebutuhan suatu sistem, yaitu sistem apa yang seharusnya digunakan. Komponen-komponen dari *use case* termasuk Aktor, *Use case*, dan Subjek (Sistem). Menurut Muslihudin &

Oktafianto (2016:65), Kumpulan simbol-simbol yang digunakan pada *use case* dapat dilihat pada tabel dibawah:

Tabel 2. 3 Simbol *Use Case*

Simbol	Nama	Deskripsi
	*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	Asosiasi	Komunikasi antara seorang aktor dengan *use case* yang terkait dengan *use case* tersebut melibatkan interkasi dengan *actor* tersebut.
	*Include*	Garis yang menggambarkan bahwa sebuah *use case* dapat menggunakan servis dari *use case* lainnya.
	*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana tambahan *use case* dapat digunakan secara mandiri.
	Generalisasi	Hubungan generalisasi dan spesialisasi antara dua buah *use case* yang pengunaan suatu fitur lebih umum dibandingkan yang lain.

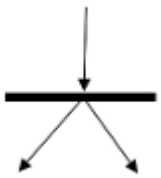
Sumber : Muslihudin & Oktafianto (2016), Analisis Dan Perancangan Sistem Informasi Menggunakan Model Terstruktur dan UML, hal.65

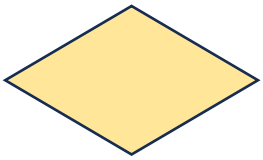
b. *Activity diagram*

Menurut Habibi et al. (2020:62), *activity* diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana setiap aliran dimulai, keputusan yang mungkin terjadi, dan bagaimana aliran tersebut berakhir. Diagram ini juga dapat menggambarkan proses paralel yang mungkin terjadi selama beberapa eksekusi.

Menurut Sari (2021:173), simbol yang dipakai dalam *activity diagram* dapat dilihat pada tabel dibawah :

Tabel 2. 4 Simbol *Activity Diagram*

Simbol	Nama	Deskripsi
	*Start point*	*Start point* direpresentasikan oleh bulatan hitam yang menandakan awal dari suatu alur kerja.
	*End point*	*End point* direpresentasikan oleh dua bulatan, dimana bulatan di dalamnya berwarna hitam yang menandakan akhir dari suatu alur kerja.
	*Activities*	Menggambarkan suatu proses aktivitas.
	*Fork*	Menunjukkan aktivitas secara paralel atau sebagai gabungan dari kegiatan paralel yang membentuk jadi satu.

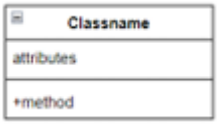
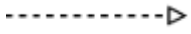
	*Join*	Untuk memperlihatkan adanya suatu uraian.
	*Decision point*	Untuk penggambaran dari pilihan antara *true* atau *false*.
	*swimlane*	*Swimlane* merepesentasikan aktor yang melakukan aktivitas.

Sumber : Sari (2021), Buku Ajar Rekayasa Perangkat Lunak, hal.173

c. *Class diagram*

Menurut Habibi et al. (2020:59), *class* adalah sebuah spesifikasi yang, ketika diinstansiasi, menghasilkan sebuah objek dan merupakan elemen utama dalam pengembangan serta desain berorientasi objek. *Class* menggambarkan kondisi (atribut/properti) dari suatu sistem, serta menyediakan layanan untuk memanipulasi kondisi tersebut (metode/fungsi). Menurut Putri & Jarti (2022:14), simbol-simbol dalam *class diagram* dapat dilihat pada tabel dibawah ini :

Tabel 2. 5 Simbol *Class Diagram*

Simbol	Nama	Deskripsi
	*Class*	*Class* merupakan kategori atau klasifikasi yang gunanya untuk menjelaskan sekumpulan yang suatu objek, digambarkan dalam bentuk persegi panjang, memiliki dua atau tiga bagian yaitu nama class, atribut, dan *method* yang dimiliki *class* tersebut.
	Asosiasi	*Association* merupakan garis penghubung antar *class*. Pada garis tersebut juga terdapat *multiplicity* yang merepresentasikan jumlah maksimum dan minimum dari satu *class* lainnya.
	*Generalization*	*Generalization* merupakan relasi yang menunjukkan hubungan antara *subclass* dengan *superclass*, di mana *subclass* merupakan bagian yang diturunkan (*inheritance*) dari *superclass*. Superclass merupakan *class* dibandingkan *subclass*.
	*Dependency*	*Dependency* merupakan relasi yang menunjukkan hubungan yang maknanya bergantung diantara *class*.

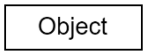
Sumber : Putri & Jarti (2022), Rancang Bangun Manajemen Akutansi Berbasis *Web Mobile*, hal.14

d. *Sequence diagram*

Menurut Habibi et al. (2020:68), *sequence* diagram menggambarkan interaksi antara objek-objek di dalam dan di sekitar sistem (termasuk pengguna, tampilan, dan lainnya) dalam bentuk pesan yang diilustrasikan seiring waktu. *Sequence diagram* memiliki dua dimensi:

vertikal (waktu) dan horizontal (objek-objek yang terlibat). Menurut Sari (2021:178), simbol-simbol dalam *sequence diagram* dapat dilihat pada tabel dibawah ini :

Tabel 2. 6 Simbol *Sequence Diagram*

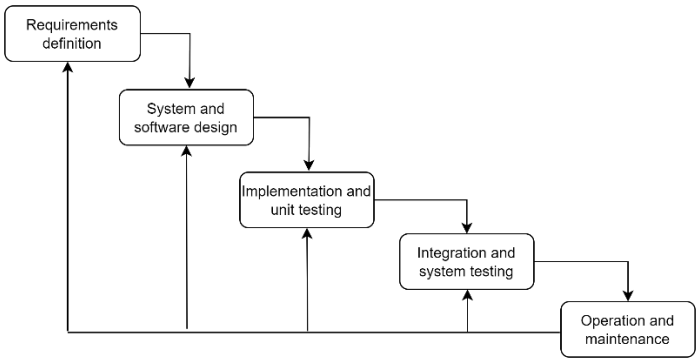
Simbol	Nama	Deskripsi
	*Object*	Komponen utama *sequence diagram*.
	*Actor*	Menggambarkan orang yang sedang berinteraksi dengan sistem.
	*Entity Class*	Menggambarkan hubungan kegiatan yang akan dilakukan.
	*Boundry Class*	Menggambarkan sebuah penggambaran dari form
	*Control Class*	Menggambarkan penghubung antara boundry dengan tabel
	*Life Line*	Menggambarkan tempat mulai dan berakhirnya sebuah *message*
	Message	Menggambarkan pengiriman pesan

Sumber : Sari (2021), Buku Ajar Rekayasa Perangkat Lunak, hal.178

7. *Waterfall*

Menurut Sadriansyah (2020:9), metode *waterfall* adalah salah satu metode pengembangan perangkat lunak yang paling sederhana dan sering disebut sebagai siklus hidup klasik. Metode ini mengusulkan pendekatan

yang sistematis dan berurutan untuk pengembangan perangkat lunak. Tahapan metode *waterfall* dapat dilihat pada gambar 2.1



Gambar 2. 1 Metode *Waterfall*

Sumber : Karnovi et al., (2020), Tutorial Membuat Aplikasi Sistem Monitoring Progres Pekerjaan Dan Evaluasi Pekerjaan Pada Job Desk Operational Human Capital Menggunakan Metode Naive Bayes, hal. 51

Menurut Irwansyah dalam (Rusmawan, 2019:90), penjelasan tahapan metode *waterfall* adalah sebagai berikut :

a. **Analisis kebutuhan**

   Proses pengumpulan kebutuhan perangkat lunak. Untuk memahami dasar dari program yang akan dibuat, seorang analis harus mengetahui ruang lingkup informasi, fungsi-fungsi yang diperlukan, kemampuan kinerja yang diharapkan, dan desain antarmuka pengguna untuk perangkat lunak tersebut.

b. **Desain sistem**

   Proses bertahap yang berfokus pada empat aspek penting yaitu struktur data, arsitektur perangkat lunak, detail prosedur, dan karakteristik antarmuka pengguna.

c. **Implementasi**

Pengodean peranti lunak merupakan proses penulisan bahasa program agar peranti lunak tersebut dapat dijalankan oleh mesin**.**

d. **Pengujian**

Proses ini menguji kode program yang telah dibuat dengan fokus pada bagian dalam perangkat lunak. Tujuannya adalah untuk memastikan bahwa semua pernyataan telah diuji dan bahwa input yang digunakan menghasilkan output yang sesuai. Pada tahap ini, pengujian dibagi menjadi dua bagian yaitu pengujian internal dan pengujian eksternal. Pengujian internal bertujuan untuk memastikan semua pernyataan telah diuji, sedangkan pengujian eksternal bertujuan untuk menemukan kesalahan dan memastikan bahwa output yang dihasilkan sesuai dengan yang diharapkan.

e. **Pemeliharaan**

Proses ini dilakukan setelah perangkat lunak digunakan oleh pengguna atau konsumen. Jika ditemukan kesalahan, perubahan akan dilakukan, sehingga perangkat lunak perlu disesuaikan kembali.

8. ***Black Box Testing***

Menurut Williams dalam (Rahmah, 2020:25), pengujian *black box*, juga dikenal sebagai pengujian fungsional, adalah pengujian yang tidak mempertimbangkan mekanisme internal sistem atau komponen dan hanya berfokus pada output yang dihasilkan sebagai respons terhadap input yang dipilih dan kondisi eksekusi. Dengan demikian, dapat disimpulkan bahwa

pengujian *black box* adalah pengujian yang berorientasi pada fungsionalitas, yaitu perilaku perangkat lunak terhadap input yang diberikan oleh pengguna untuk mendapatkan output yang diinginkan tanpa memperhatikan proses internal atau kode program yang dijalankan oleh perangkat lunak.

Menurut Kusuma et al. (2024:184), pengujian *black box* berupaya untuk menemukan kesalahan dalam kategori berikut :

a. Fungsi yang salah atau hilang
b. Kesalahan antarmuka
c. Kesalahan dalam struktur data atau akses baris data eksternal
d. Kesalahan perilaku atau kinerja
e. Kesalahan inisialisasi dan penghentian

Menurut Taufik dalam (Kusuma et al., 2024:185), ciri-ciri *black box* adalah sebagai berikut :

a. *Black box testing* berfokus pada kebutuhan fungsional perangkat lunak, didasarkan pada spesifikasi kebutuhan perangkat lunak tersebut.
b. *Black box testing* dilakukan tanpa mengetahui secara detail struktur internal dari sistem atau komponen yang diuji. Pengujian ini juga dikenal sebagai pengujian perilaku, pengujian berbasis spesifikasi, pengujian input/output, atau pengujian fungsional.

B. Kajian Empiris

Terdapat bahasan yang dibahas terkait dengan perancangan aplikasi presensi karyawan banyak publikasi dalam bentuk jurnal ilmiah. Dalam penelitian Perancangan Aplikasi Presensi Karyawan Menggunakan *Face Recognition* (Studi Kasus Pada Teh Kota) perlu beberapa penelitian dari beberapa sumber yang cocok untuk menjelaskan terkait aplikasi presensi karyawan berbasis *mobile*.

Penelitian yang telah dilakukan oleh Hidayah & Saifudin (2023) bahwa peneliti tersebut melakukan penelitian dengan merancang dan membangun sistem presensi menggunakan metode *waterfall* .Batasan dalam menampilkan menu *home*, halaman, menu data sebelum presensi, menu sesudah presensi, tampilan seluruh data presensi dengan menerapkan metode penguji sistem dimana menggunakan metode *black box* dan metode *white box*. Metode *black box* nantinya akan diterapkan pada tampilan fungsional pengguna, apakah ada kesalahan berupa eror atau tidak, sedangkan metode *white box* diterapkan pada struktur dalam kodingan pada aplikasi pada aplikasi, apakah kode sudah sesuai atau belum.

Penelitian yang dilakukan oleh Erzed et al. (2022), dengan menggunakan tahapan penelitian dimulai dari analisa sistem berjalan, perancangan sistem, *prototyping*, dan pengujian. Terdapat permasalahan yang terjadi dalam penelitian ini adalah PT.Railink dalam kesulitan presensi karyawan di lapangan dan pengelolaan presensi karyawan yang masih menggunakan metode konvensional. Hal ini dapat menjadi tidak efesien dan tidak akurat terutama

untuk karyawan yang bekerja di lapangan atau tidak dalam satu area kerja. Sehingga dirancang aplikasi presensi karyawan berbasi *mobile* yang diharapkan masalah-masalah terebut dapat diatasi dan kinerja perusahaan terutama dalam pengelolaan presensi karyawan dapat ditingkatkan.

Penelitian yang dilakukan oleh Ulumudin et al. (2023), dengan menggunakan metode penelitian yaitu metode *waterfall* dengan beberapa tahapan antara lain pengumpulan kebutuhan, desain, implementasi, pengujian, instalasi, dan pemeliharaan. Permasalahan yang dihadapi dalam mengimplementasikan presensi masih beruba tanda tangan untuk pegawai. Kehadiran setiap pegawai dicatat melalui sistem presensi yang terkadang ada pegawai melakukan kecurangan seperti titip presensi dan kesulitan memantau presensi secara *real-time.* Sistem untuk pengelolaan presensi di PT.Berkah Pena Ilmu masih dilakukan secara manual, sehingga dirasa kurang efesien dan efektif. Sehingga diciptakan sebuah sistem yaitu aplikasi presensi pegawai PT.Berkah Pena Ilmu berbasis *Android* menggunakan *firebase* bertujuan untuk mengurangi kecurangan dengan menggunakan teknologi *location based service* (LBS) dan memudahkan pemantau presensi secara *real-time.*

Penelitian yang dilakukan Afrianto & Lubis (2023), dengan menggunakan metode pengembangan perangkat lunak yang digunakan adalam *system development life cycle* (SDLC) dengan tahapan perencanaan, analisis, perancangan, implementasi, dan pengujian. Permasalahan yang dihadapi dalam sistem absensi menggunakan aplikasi *whatsapp* dengan foto wajah dan karyawan yang bekerja dari rumah menyebabkan ketidakstabilan dalam sistem

absensi yang berlaku . Hal ini kemudian akan menyebabkan tingkat akurasi dan efektif kurang karena banyak karyawan yang bekerja dari rumah. Sehingga dirancang aplikasi absensi karyawan Universitas Harapan Medan menggunakan *face recognition* untuk meningkatkan akurasi dan efektivitas dalam proses absensi.

Penelitian yang dilakukan Wijaya et al. (2023), dengan menggunakan metode perancangan yaitu metode CNN. Permasalahan yang terjadi yaitu ketidakpraktisan sistem presensi yang ada dan ketidakakuratan dan ketidakefesienan sistem presensi menggunakan fingerprint dan scan QR dianggap tidak praktis karena dapat menimbulkan antrean yang panjang atau memerlukan waktu. Sehingga menghasilkan perancangan aplikasi sistem presensi pegawai Universitas Atma Jaya Yogyakarta dengan *face recognition* yang memumdahkan untuk meningkatkan akurasi, efesiensi, dan kemudahan dalam proses presensi.

Penelitian yang dilakukan oleh Pipin (2024), dengan menggunakan metode perancangan yaitu *waterfall* yang terdiri dari pengumpulan data, analisis proses, analisi kebutuhan, perancangan dan implementasi. Permasalahn yang terjadi adalah sistem presensi di CV.Global Mandiri saat ini menggunakan pancatatan kertas yang rentan terhadap praktik kecurangan dan manipulasi kehadiran karyawan sehingga memakan waktu dan tidak efesien. Hal ini menyebabkan ketidakpastian kehadiran karyawan dalam menilai kedisiplinan karyawan. Sehingga diciptakan sebuah sistem yaitu aplikasi presensi *online* karyawan CV.Global Mandiri berbasis *mobile* dengan penerapan *geolacator* dan *face*

recognition bertujuan untuk pencegahan terhadap praktik kecurangan dengan penggunaan *plugin* untuk mendeteksi lokasi dan memverifikasi kebenaran karyawan.

Pengembangan yang dilakukan ke dalam sistem yaitu dengan dibuatnya sistem aplikasi presensi karyawan dengan menggunakan *face recognition* yang dilengkapi dengan fitur riwayat presensi, rekap presensi, jam terlambat, lembur, rekap gaji guna untuk membantuk owner dalam menginputkan gaji karyawan berdasarkan gaji perjam, potongan gaji untuk terlambat, dan tambahan jika ada lemburan.

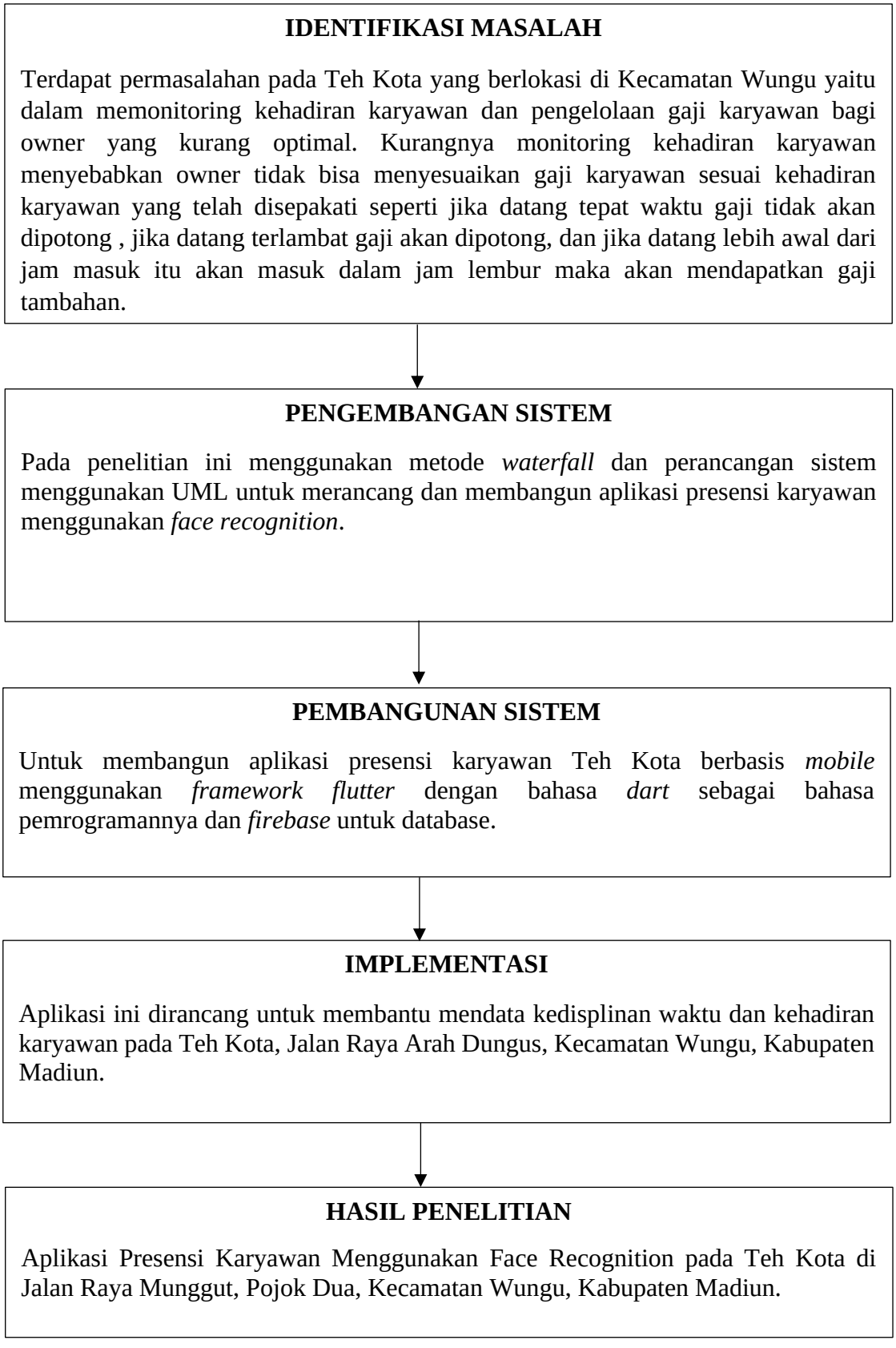
C. Kerangka Berpikir

Teh Kota merupakan sebuah bisnis minuman es teh kekinian yang berasal dari Blora, Jawa Tengah yang menerapkan sistem *franchise* yang terletak di Jalan Raya Munggut, Pojok Dua, Kecamatan Wungu. Terdapat kurangnya belum memiliki aplikasi presensi untuk mendata kehadiran karyawan sehingga mempersulit owner untuk melihat ketepatan waktu karyawan saat masuk dan pulang bekerja.

Pada penelitian yang dilakukan ini menggunakan model pengembangan yaitu metode *waterfall*. *Waterfall* adalah model yang memiliki konsep pendekatan seperti alur hidup perangkat lunak yang terjadi secara sekuensial atau terurut dimulai dari analisis, desain, pengodean serta pengujian. Tahapan-tahapan dari metode *waterfall* antara lain analisis, desain sistem, implementasi, pengujian, pemeliharaan.

Aplikasi presensi karyawan Teh Kota yang berbasis mobile ini dibangun menggunakan teknologi *face recognition* dengan menggunakan pengembangan *framework flutter* dengan bahasa *dart* sebagai bahasa pemrogramannya yang terintegrasi dengan layanan *backend* basis data *firebase*. Uji kelayakan kualitas pada sistem dilakukan oleh peneliti menggunakan pengujian *black box*. Aplikasi presensi karyawan yang akan diimplementasikan pada presensi karyawan Teh kota terdapat presensi masuk, presensi pulang, karyawan dapat melihat jam lembur, riwayat presensi , dan rekap presensi.

Hasil pada penelitian ini adalah Aplikasi Presensi Karyawan Berbasis Mobile pada Teh Kota yang berada di Jalan Raya Munggut, Pojok Dua, Kecamatan Wungu, Kabupaten Madiun. Dalam aplikasi presensi yang dibangun ini dapat membantu owner dalam menentukan gaji karyawan sesuai rekap presensi dan melihat riwayat kehadiran karyawan. Berdasarkan uraian tersebut dapat dibuat sebuah kerangka berpikir yang telah disusun oleh peneliti yang dijelaskan dalam gambar 2.2 sebagai berikut :



Gambar 2. 2 Kerangka Berpikir