RANCANG BANGUN APLIKASI PRESENSI DENGAN METODE PENGENALAN WAJAH MENGGUNAKAN FACENET

TUGAS AKHIR

NATASYA LIEMENA

311910012

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS MA CHUNG

MALANG

2023

LEMBAR PENGESAHAN

TUGAS AKHIR

RANCANG BANGUN APLIKASI PRESENSI DENGAN METODE PENGENALAN WAJAH MENGGUNAKAN FACENET

Oleh:

NATASYA LIEMENA

NIM. 311910012

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS dan TEKNOLOGI

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana Komputer (S.Kom.)

Dosen Pembimbing I,	Dosen Pembimbing II,
Windra Swastika, S.Kom., MT., Ph.D.	**Ir. Oesman Hendra Kelana, M.Div., M.Cs.**
NIP. 20070039	**NIP. 20110022**

Dekan Fakultas Sains dan Teknologi,

Dr. Kezia Rega Prilianti, M.Si.

NIP. 20120035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan "Rancang Bangun Aplikasi Presensi dengan Metode Pengenalan Wajah Menggunakan FaceNet" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 20 Juli 2023

Natasya Liemena
NIM. 311910012

RANCANG BANGUN APLIKASI PRESENSI DENGAN METODE PENGENALAN WAJAH MENGGUNAKAN FACENET

Natasya Liemena, Windra Swastika, Oesman Hendra Kelana

Abstrak

Pada tahun 2022 di mana pandemi telah berjalan selama lebih dari dua tahun mengharuskan karyawan membagi jadwal kerja dengan *hybrid working*. Hal ini menyebabkan perusahaan dengan ribuan karyawan memiliki kesulitan untuk mendata presensi setiap karyawannya. Oleh karena itu, dibutuhkan sistem yang dapat membantu pendataan presensi karyawan. Pada penelitian ini akan membuat aplikasi *mobile* dengan pengenalan wajah dan *spoofing detection* untuk membantu proses pendataan presensi karyawan.

Penelitian ini akan menggunakan FaceNet sebagai model untuk melakukan pengenalan wajah. Selain pengenalan wajah juga dibutuhkan pendeteksi wajah dan *spoofing detection* menggunakan BlazeFace. Hal ini agar aplikasi dapat mendeteksi wajah yang ingin diidentifikasi dan juga mencegah kecurangan presensi dengan menggunakan foto pengguna. Pengembangan aplikasi pada penelitian ini menggunakan Flutter agar aplikasi dapat dijalankan di platform iOS dan Android.

Aplikasi presensi dengan pengenalan wajah dan *spoofing detection* yang digunakan untuk mengenali wajah dan mendata kehadiran dapat berjalan dengan baik pada platform Android dan iOS. Tingkat akurasi yang dimiliki untuk *face recognition* yaitu sebesar 83.33%, sedangkan tingkat akurasi yang dimiliki untuk *spoofing detection* yaitu sebesar 100%.

Kata Kunci : *BlazeFace, FaceNet, Face Recognition, Flutter, Spoofing Detection*

APLICATION DEVELOPMENT WITH FACE RECOGNITION USING FACENET

Natasya Liemena, Windra Swastika, Oesman Hendra Kelana

Abstract

In the year of 2022 where pandemic has been going for more than two years forced employees to divide their work schedule with hybrid working. This causes companies with thousands of employees to have difficulty recording the attendance of each employee. Therefore, a system is nedeed that can help record employee attendance. This study will create a mobile application with facial recognition dan spoofing detection to help employee attendance data collection process.

The research will use FaceNet as a model to perform facial recognition. In addition to face recognition, face detection dan spoofing detection using BlazeFace are also needed. This is so that the application can detect the face you want to identify and also prevent attendance fraud by using the user's photo. Application development in this study uses Flutter so that the application can run on both iOS and Android platforms.

Attendance application with facial recognition and spoofing detection used to recognize faces and record attendance data can run well on both Android and iOS platforms. The accuracy rate for facial recognition is 83.33%, while the accuracy rate for spoofing detection is 100%.

Key Words : BlazeFace, FaceNet, Face Recognition, Flutter, Spoofing Detection

KATA PENGANTAR

Puji syukur dipanjatkan kehadirat Tuhan Yang Maha Esa, yang atas restu-Nya sehingga tugas akhir dengan judul "Rancang Bangun Aplikasi Presensi dengan Metode Pengenalan Wajah Menggunakan FaceNet" dapat diselesaikan dengan sebaik-baiknya. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada seluruh pihak yang telah membantu penulis dalam proses pembuatan tugas akhir, di antaranya:

1. Ibu Dr.Kestrilia Rega Prilianti, M.Si selaku Dekan dari Fakultas Sains dan Teknologi Universitas Ma Chung,
2. Bapak Hendry Setiawan, ST, M.Kom, selaku Kepala Program Studi Teknik Informatika dan Dosen Penguji Tugas Akhir,
3. Bapak Windra Swastika, S.Kom., MT., Ph.D, selaku Dosen Pembimbing I Tugas Akhir,
4. Bapak Ir. Oesman Hendra Kelana, M.Div, M.Cs, selaku Dosen Pembimbing II Tugas Akhir,
5. Kedua orang tua terkasih, yang telah memberikan dukungan dan semangat selama proses pembuatan Tugas Akhir,
6. Serta teman-teman yang telah memberikan dukungan untuk dapat menyelesaikan Tugas Akhir,

Laporan ini disusun berdasarkan hasil proyek tugas akhir dengan judul "Rancang Bangun Aplikasi Presensi dengan Metode Pengenalan Wajah Menggunakan FaceNet" sebagai salah satu prasyarat kelulusan. Penulis menyadari bahwa tugas akhir ini masih memiliki kekurangan dan jauh dari kata sempurna. Oleh karena itu kritik juga saran sangat membantu memperbaiki kekurangan yang ada. Demikian tugas akhir ini dapat bermanfaat.

Malang, 20 Juli 2023

Natasya Liemena

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada tahun 2022 di mana pandemi telah berjalan selama lebih dari dua tahun mengharuskan masyarakat untuk membatasi bepergian dan kontak dengan orang lain. Hal ini juga telah mempengaruhi pola kerja karyawan kantor yang mengharuskan karyawan membagi jadwal kerja untuk meminimalisir kontak dekat dengan orang lain. *Hybrid working* adalah penggabungan metode kerja di mana karyawan bekerja di kantor *Work From Office* (WFO) dan dari jarak jauh *Work From Home* (WFH) (Chaf, Hultberg, & Yams, 2022).

Perusahaan-perusahaan besar di Indonesia memiliki ribuan karyawan, dan dengan diberlakukannya *hybrid working*, karyawan tidak selalu bekerja dari kantor. Ketika di kantor, karyawan biasanya melakukan presensi dengan mesin absen yang memerlukan sidik jari. Namun ketika bekerja di luar kantor tidak ada fasilitas seperti mesin absen sidik jari yang dapat membantu karyawan untuk absen. Pilihan lain seperti menggunakan form online memiliki banyak kekurangan. Kekurangan yang pertama adalah tidak dapat diketahui secara pasti siapa yang mengisi form tersebut, sehingga tidak dapat mengetahui kecurangan di mana form seseorang diisi oleh orang lain. Selain itu dengan jumlah yang besar, pengisian presensi melalui form menambah kesulitan untuk mendata kehadiran setiap karyawannya. Oleh karena itu, untuk mempermudah pendataan presensi, diperlukan aplikasi yang dapat membantu mendata kehadiran karyawan secara efisien sekaligus meminimalisir adanya tindak kecurangan dalam pendataan presensi.

Aplikasi yang dikembangkan memerlukan fitur yang dapat mengenali wajah karyawan untuk memastikan bahwa yang melakukan presensi adalah karyawan itu sendiri, sehingga meminimalisir terjadinya kecurangan absen oleh orang lain. Sistem Pengenalan wajah adalah program komputer yang secara otomatis menggunakan gambar digital atau bingkai video dari sumber video untuk mengidentifikasi atau mengautentikasi seseorang secara otomatis (Kumaran, et al., 2022). Pengenalan wajah akan menggunakan model FaceNet yang dikembangkan

oleh peneliti dari Google. Selain itu juga akan ditambahkan proses *spoofing detection*, untuk mencegah terjadinya kecurangan di mana pengguna akan melakukan presensi dengan menggunakan foto. *Spoofing detection* yang dilakukan adalah meminta pengguna untuk melakukan instruksi acak seperti menghadap ke atas, bawah, kiri, dan kanan.

Kendala lain adalah pengguna yang menggunakan produk dari Apple seperti iPhone dan iPad semakin banyak dengan semakin berkembangnya perusahaan Apple. Hal ini mengakibatkan bertambahnya pengguna yang menggunakan sistem operasi iOS. Meskipun sistem operasi Android masih mendominasi pasar, namun pengguna iOS yang semakin berkembang menjadi pertimbangan *developer* ketika mengembangkan aplikasi agar aplikasi yang dibuat dapat digunakan oleh banyak orang. Oleh karena itu, untuk mengembangkan aplikasi yang dapat digunakan oleh berbagai platform, aplikasi absen akan dibuat sebagai aplikasi *multi platform* yang menggunakan *framework* Flutter.

Dengan aplikasi yang menggunakan *face recognition* diharapkan dapat membantu membuat proses pendataan karyawan untuk menjadi lebih efisien dan optimal. Juga dengan pengembangan aplikasi yang dapat digunakan pada *multi-platform* memberikan kebebasan dan jangkauan yang luas bagi pengguna dengan berbagai sistem operasi.

1.2 Identifikasi Masalah

Identifikasi masalah dalam penelitian ini adalah proses presensi yang sudah berjalan tidak dapat diaplikasikan pada metode kerja *hybrid working* sehingga diperlukan suatu aplikasi presensi secara online dengan sistem *face recognition* untuk membantu proses presensi ketika *hybrid working*.

1.3 Batasan Masalah

Batas masalah dalam tugas akhir ini adalah sebagai berikut :

a. Aplikasi yang dikembangkan memasukkan data presensi masuk dan pulang kerja.
b. Pengembangan aplikasi yang dikerjakan bagian *front-end* dan *back-end*.

c. Pengembangan apikasi pengenalan wajah menggunakan model FaceNet dan *spoofing detection* dengan BlazeFace.
d. Aplikasi tidak melakukan pengujian dengan aplikasi *mock location*.

1.4 Rumusan Masalah

Rumusan masalah dalam tugas akhir ini adalah bagaimana merancang aplikasi presensi karyawan selama *hybrid working* dengan menggunakan metode mengenali wajah.

1.5 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah membuat aplikasi presensi *multi platform* dengan *face recognition* untuk membantu karyawan melakukan presensi selama *hybrid working*.

1.6 Manfaat Penelitian

Manfaat penelitian yang dapat diperoleh dari tugas akhir ini adalah :

a. Bagi pembaca, dapat digunakan sebagai referensi untuk belajar dan pengembangan aplikasi *multi-platform.*
b. Bagi Universitas Ma Chung, khususnya program studi Teknik informatika dapat membekali mahasiswa dengan pengalaman dalam pengerjaan tugas akhir ini serta menjalin hubungan baik dengan perusahaan yang memiliki proyek ini.
c. Bagi penulis, dapat menerapkan ilmu serta memperluas wawasan terkait pengembangan aplikasi *mobile* yang diperoleh di Universitas Ma Chung serta ketika mengikuti kegiatan magang.

1.7 Luaran Tugas Akhir

Luaran dari tugas akhir ini adalah aplikasi absen *multi-platform* dengan sistem *face recognition* dilengkapi dengan *spoofing detection* yang dapat digunakan untuk proses presensi dan hasil akan dipublikasikan dalam bentuk jurnal.

1.8 Sistematika Penulisan

Sistematika laporan penelitian tugas akhir ini dibagi dalam lima bab sebagai berikut :

1. Bab 1 Pendahuluan

   Pada Bab Pendahuluan berisikan mengenai latar belakang tugas akhir ini, identifikasi masalah, batasan dari masalah yang diidentifikasi, rumusan masalah, tujuan penelitian tugas akhir ini dilaksanakan, manfaat dari penelitian tugas akhir bagi berbagai pihak, sitematika dari penulisan tugas akhir, dan rencana penelitian tugas akhir.

2. Bab 2 Tinjauan Pustaka

   Pada Bab Tinjauan Pustaka, dijelaskan mengenai teori yang melandasi penelitian tugas akhir ini, seperti Flutter, Dart, Firebase, *Machine Learning Kit*, FaceNet, *Cosine Similarity*, BlazeFace, Tensor Flow Lite, Java, Hive, *Database*, Postgre SQL, *Web Service*, REST API, Visual Studio Code, Netbeans, serta penelitian terdahulu. Teori yang dipaparkan bersumber dari buku, jurnal, serta *website* yang berkaitan.

3. Bab 3 Metode Penelitian

   Pada Bab Metode Penelitian, dijelaskan mengenai perancangan sistem dari aplikasi presensi dengan *face recognition* yang dikembangkan.

4. Bab 4 Hasil dan Pembahasan

   Pada Bab Hasil dan Pembahasan, terdapat penjelasan terkait hasil dari pengembangan aplikasi serta pengujian yang dilakukan. Dari pengujian yang dilakukan akan dipaparkan tingkat keberhasilan, kritik, serta saran sebagai evaluasi dari aplikasi yang dikembangkan.

5. Bab 5 Kesimpulan dan Saran

   Pada Bab Kesimpulan dan Saran, terdapat kesimpulan dari penelitian tugas akhir yang dilakukan serta saran untuk penelitian berikutnya yang akan mengembangkan hasil penelitian ini kedepannya.

BAB II
TINJAUAN PUSTAKA

2.1 Flutter

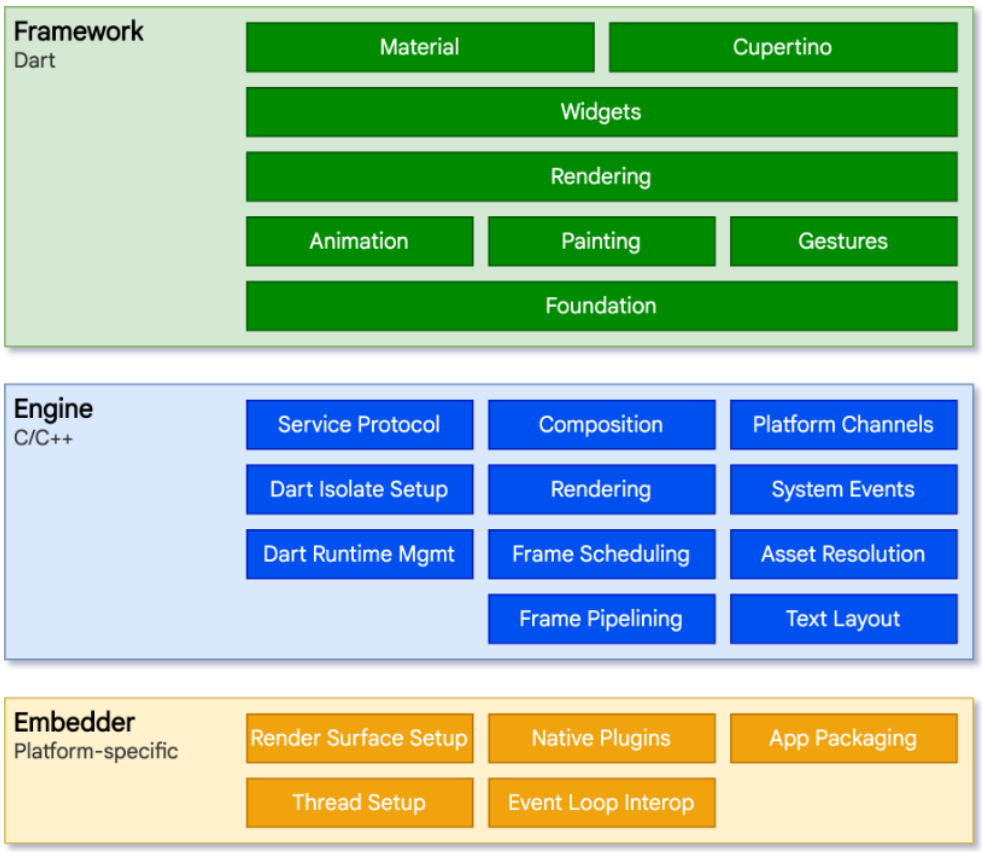
Flutter merupakan platform yang digunakan untuk mengembangkan aplikasi *multiplatform* dengan satu *codebase*, di mana aplikasi ini dapat digunakan di Android, iOS, web, maupun desktop. Flutter telah dikembangkan oleh Google sejak tahun 2015, dan secara resmi diluncurkan pada Desember 2018. Sejak diluncurkan, flutter telah meraih popularitas yang tinggi dan digunakan oleh banyak orang. Flutter merupakan platform yang *open-source*, di mana Flutter dapat digunakan gratis oleh semua orang.

Flutter menggunakan bahasa Dart. Sama seperti Flutter, bahasa Dart juga dikembangkan oleh Google dan merupakan bahasa pemrograman resmi bagi Flutter. Bahasa ini dirancang agar familiar dengan bahasa lainnya untuk memudahkan transisi bagi pengembang yang telah menguasai bahasa lain. Flutter akan mengompilasi kode ke ARM, mesin Interl, atau JavaScript untuk menghasilkan performa yang cepat.

Flutter memiliki kelebihan untuk mengembangkan aplikasi dengan lebih mudah dan cepat. Kelebihan ini didukung dengan adanya fitur *widgets* yang dapat diubah sesuai keinginan dengan mudah dan juga fitur *Hot Reload* yang memudahkan proses pengembangan aplikasi. Namun aplikasi yang dihasilkan Flutter lebih besar dan sedikit lebih lambat dibandingkan dengan aplikasi *native*.

Gambar 2.1 menunjukkan Flutter yang dirancang sebagai sistem dengan lapisan-lapisan yang dapat diperluas. Flutter memiliki serangkaian *library*, di mana masing-masing *library* ini bergantung pada lapisan yang mendasarinya. Setiap bagian dari lapisannya dirancang sehingga bersifat optional dan dapat digantikan.

Sistem operasi yang mendasari Flutter dikemas dengan cara yang sama seperti aplikasi *native* lainnya. *Embedder* platform spesifik memberikan titik masuk dan berkoordinasi dengan sistem operasi dasar untuk akses ke layanan seperti *rendering surfaces*, aksesbilitas, dan *input*, juga dapat mengelola perputaran *message event*.



Gambar 2.1 Arsitektur Flutter (Google, 2023)

Embedder ini ditulis dalam bahasa yang sesuai untuk tiap platform, Java dan C++ untuk Android, Objective-C atau Objective-C++ untuk iOS dan macOS, juga C++ untuk Windows dan Linux. Dengan adanya *embedder*, kode Flutter dapat diintegrasikan menjadi aplikasi yang ada sebagai modul, atau kode tersebut berupa seluruh konten aplikasi. Flutter menyertakan sejumlah *embedder* untuk platform yang umum, tetapi ada juga embedder lain.

Inti dari Flutter adalah Flutter *engine*, yang sebagian besar ditulis dalam C++ dan mendukung kebutuhan primitive untuk menyokong seluruh aplikasi Flutter. *Engine* bertanggung jawab untuk merasterisasi adegan gabungan ketika *frame* baru perlu dibuat. *Engine* menyediakan implementasi tingkat rendah untuk *core* API Flutter, termasuk grafik melalui SKIA, *text layout*, *file* dan *network I/O*, *accessibility support*, *plugin architecture*, dan *runtime* juga *compile toolchain* Dart. *Engine* terhubung ke *framework* Flutter melalui dart:ui, yang membungkus kode

C++ yang mendasarinya di kelas Dart. Library ini memaparkan primitif tingkat terendah, seperti kelas untuk *driving input*, grafik, dan subsistem perenderan teks.

Pada umumnya, *developer* akan berinteraksi dengan Flutter melalui *framework* Flutter. Di mana *framework* Flutter ini menyediakan *framework* modern, reaktif yang ditulis dalam bahasa Dart. *Framework* mencakup serangkaian *platform*, *layout*, dan *foundational libraries*, terdiri dari serangkaian lapisan.

Berikut bagian-bagian dari *framework* Flutter dari bawah ke atas :

a. Lapisan *foundation*, terdapat kelas dasar dan *building block services* seperti animasi, gambar, dan gerak yang menawarkan abstraksi umum untuk digunakan di atas *underlying foundation*.
b. Lapisan *rendering*, lapisan ini menyediakan abstraksi untuk menangasi *layout*. Dengan lapisan ini, *developer* dapat membuat pohon objek yang dapat direnderi. Objek ini dapat dimanipulasi secara dinamis, dengan pohon yang secara otomatis akan memperbarui *layout* untuk mencerminkan perubahan-perubahan dari *developer*.
c. Lapisan *widget*, lapisan ini adalah abstraksi komposisi. Di mana setiap objek render di lapisan *rendering* memiliki kelas yang sesuai di lapisan *widget*. Lapisan ini juga memungkinkan untuk menentukan kombinasi kelas yang dapat digunakan kembali. Ini adalah lapisan di mana diperkenalkan model pemrograman reaktif.
d. Lapisan material dan cupertino, lapisan *library* ini memberikan serangakaian kontrol komprehensif yang menggunakan komposisi primitif lapisan *widget* untuk mengimpementasi bahasa desain Material atau iOS.
e. *Framework* Flutter relatif kecil, banyak fitur tingkat tinggi yang mungkin digunakan *developer* diimplementasi sebagai packages, termasuk platform plugin seperti kamera dan tampilan web, juga fitur *platform-agnostic* seperti katakter, HTTP, dan animasi yang dibangun atas *library* inti Dart dan Flutter. Sebagian dari *package* ini merupakan bagian ekosistem yang lebih besar, beberapa contohnya adalah layanan seperti autentikasi Apple, pembayaran dalam aplikasi, dan animasi.

Seiring dengan perkembangan perangkat keras yang semakin lama memiliki berbagai fitur baru, versi Flutter juga terus ditingkatkan untuk dapat memanfaatkan komponen dan fitur yang terus berkembang.

2.2 Dart

Dart merupakan bahasa pemrograman resmi untuk *framework* Flutter yang dikembangkan oleh Google. Dart adalah bahasa yang *object-oriented*, *class-based*, *garbage-collected* dengan *C-style syntax*. Pada Dart, segala sesuatu yang ditempatkan dalam suatu variabel merupakan objek, dan setiap object merupakan turunan dari kelas. Meskipun Dart diketik dengan kuat, *type annotations* opsional karena Dart dapat menyimpulkan tipe. Dart memiliki pengamanan *null* yang jika diaktifkan maka variabel tidak dapat berisikan *null* kecuali memang dideklarasikan dapat berisi *null*. Dart dapat mendukung tipe yang generik dan juga fungsi tingkat atas (seperti *main()*), serta fungsi yang berkaitan dengan *class* atau *object*. Dart mendukung variabel tingkat atas, serta variabel yang terikat pada *class* atau *object*. Bahasa Dart tidak memiliki kata kunci *public*, *private,* dan *protected* seperti yang ada pada bahasa Java. Jika suatu pengidentifikasi dimulai dengan garis bawah, maka hal tersebut bersifat pribadi untuk *library*-nya.

Tool Dart dapat melaporkan dua jenis masalah, yaitu peringatan dan *error*. Peringatan hanyalah indikasi bahwa kode yang ditulis mungkin tidak berfungsi, tetapi tidak akan mencegah eksekusi programnya. *Error* dapat terjadi ketika waktu kompilasi atau waktu proses. *Error* pada waktu kompilasi mencegah kode untuk dieksekusi sama sekali, sedangkan *error* pada waktu proses akan menghasilkan *exception* ketika kode dieksekusi.

Beberapa tipe yang didukung oleh Dart adalah *int, double, string, bool, list, set, map, runes, symbol,* dan *null*. Beberapa tipe lain yang juga memiliki peran spesial pada bahasa Dart adalah *object, enum, future dan stream, iterable, never, dynamic,* dan *void*. Contoh untuk mendeklarasikan variabel dalam bahasa Dart :

```
var name = 'Budi';
String name = 'Budi';
```

Fungsi pada Dart merupakan objek dan memiliki tipe karena Dart merupakab bahasa yang berorientasi pada objek. Oleh karena itu, fungsi dapat diberikan pada variabel atau dikirim sebagai argument pada fungsi lainnya. Contoh dalam membuat fungsi pada Dart :

```
bool isBigger(int num1, int num2) {
  return num1 > num2;
}
```

2.3 Firebase

Firebase adalah salah satu dari layanan Google yang dirancang untuk memudahkan *developer* dalam pengembangan aplikasi. Firebase atau Backend as a Service (BaaS) membantu agar *developer* aplikasi tidak perlu memberikan usaha yang besar dalam masalah *backend*. Firebase awal didirikan oleh Andrew Lee sera James Tamplin pada tahun 2011. Produk ini digunakan untuk menyimpan dan sinkronisasi data ke banyak pengguna. Google mengakusisi Firebase pada tahun 2014. Dalam firebase terdapat dua jenis layanan, SPARK yang gratis dan BLAZE yang dikenakan biaya setiap bulan sesuai penggunaan layanannya.

Berikut beberapa contoh fitur dari Firebase :

a. *Firebase Analytics,* fitur ini dapat digunakan untuk koleksi data dan laporan bagi aplikasi.

b. *Firebase Cloud Messaging and Notifications*, fitur ini dapat digunakan untuk mengirimkan pesan serta notifikasi antar pengguna. Pesan notifikasinya sendiri terintegrasi dengan Google Analytics for Firebase sehingga pengguna dapat mengakses interaksi dan *tracking* konversi dengan detail.

c. *Firebase Authentication*, fitur ini digunakan untuk melakukan autentikasi penyedia identitas gabungan seperti Google, Facebook, dan lainnya atau juga dengan kata sandi dan nomor telepon.

d. *Firebase Cloud Firestore*, basis data NoSQL di *cloud* yang dapat diakses menggunakan SDK oleh aplikasi dari berbagai platform, seperti iOS, web, dan juga Android.

e. *Firebase Realtime Database*, merupakan *database cloud* yang disimpan dan dieksekusi dalam bentuk JSON, datanya disinkronisasi secara *realtime* pada setiap pengguna.

f. *Firebase Hosting*, fitur ini dapat digunakan untuk menampilkan, mengirimkan, dan mendukung berbagai jenis konten untuk di-*hosting*.

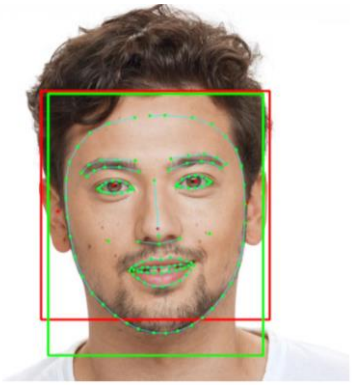
2.4 Machine Learning Kit (ML Kit)

ML Kit merupakan SDK *mobile* yang mendukung penggunaan Machine Learning di aplikasi iOS dan Android. ML Kit menyediakan API yang dapat digunakan untuk membantu *developer* menggunakan model TensorFlow Lite di aplikasi yang dikembangkan. Beberapa API yang dapat digunakan adalah pengenalan *landmark*, pengenalan teks, pemindaian *barcode*, deteksi wajah, serta melabel gambar. API perangkat akan memproses data dan dapat berjalan tanpa koneksi jaringan, sementara API *cloud* akan menggunakan teknologi *machine learning* milik Google Cloud Platform untuk memberikan akurasi yang lebih tinggi.

Selain menggunakan API yang sudah tersedia, pengembang dapat menerapkan model TensorFlow Lite dengan mengunggahnya ke Firebase *console*. Dengan ini pengembang dapat mengurangi besar aplikasi mereka. Model yang ada di *cloud* juga dapat diganti secara dinamis, sehingga tidak perlu *update* aplikasi hanya untuk mengganti modelnya.

2.5 BlazeFace

BlazeFace adalah model *machine learning* yang dikembangkan oleh Google. Dengan semakin banyak ragam perbaikan dalam arsitektur deep networks yang memungkinkan untuk deteksi objek secara *real-time*. Terutama dalam aplikasi seluler di mana hal tersebut merupakan bagian pertama dalam pemrosesan video. Oleh sebab itu dibutuhkan model yang dapat mendeteksi secepat mungkin. BlazeFace dikembangkan untuk mendeteksi posisi dan fitur dari wajah dengan cepat. Terdapat enam fitur yang dideteksi oleh BlazeFace, kedua mata, hidung, kedua telinga, dan mulut. BlazeFace juga memungkinkan untuk mendeteksi beberapa wajah sekaligus. Gambar 2.2 menunjukkan contoh hasil dari BlazeFace.



Gambar 2.2 Hasil BlazeFace (Bazarevsky, Kartynnik, Vakunov, Raveendran, & Grundmann, BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs, 2019)

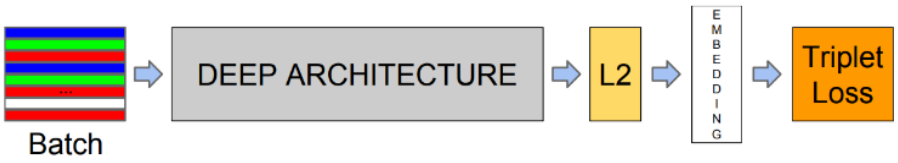
BlazeFace menggunakan jaringan yang lebih baik berdasarkan MobileNet. Di mana penulis mengganti 3x3 *depthwise convolution* menjadi 5x5 *depthwise convolution* untuk membuat model yang lebih dangka dan mempercepat prosesnya. BlazeFace dilatih dengan *dataset* yang terdiri atas 66.000 gambar. Untuk evaluasi, digunakan kumpulan data pribadi sebanyak 2.000 gambar yang beragam secara geografis.

2.6 FaceNet

FaceNet merupakah suatu sistem pengenalan wajah yang dikembangkan oleh Google *Researchers* pada tahun 2015. FaceNet mempelajari pemetaan dari gambar wajah menjadi ruang Euclidean di mana jarak berhubungan langsung dengan kemiripan wajah. Setelah menciptakan ruang tersebut, hal-hal seperti *face recognition*, *verification*, dan *clustering* dapat dengan mudah diimplementasikan dengan teknik standar dengan menjadikan FaceNet sebagai vektor fitur (Schroff, Kalenichenko, & Philbin, 2015). FaceNet hanya menggunakan 128 bit per wajah. Pada data Labeled Faces in the Wild (LFW), sistem FaceNet menjadi akurasi sebesar 99,63%. Dan pada Youtube Faces DB memiliki akurasi sebesar 95,12%.

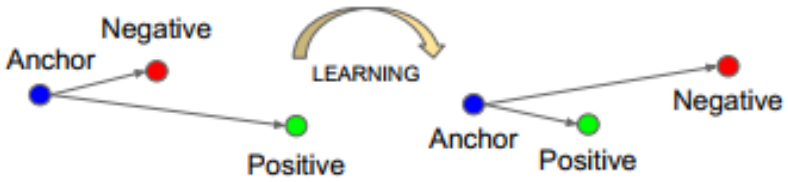
FaceNet menggunakan pembelajaran *end-to-end* dan ZF-Net atau Inception Network sebagai dasar dari arsitekturnya. Model deep learning menampilkan embedding dari gambar f(x) dengan normalisasi L_2. Kemudian diteruskan ke fungsi *loss* untuk menghitung kerugiannya. Tujuan dari fungsi ini adalah untuk membuat jarak kuadrat dari dua embedding gambar dari identitas yang sama menjadi kecil,

sedangkan jarak kuadrat dari dua *embedding* gambar dari identitas yang berbeda menjadi besar. Oleh sebab itu, digunakanlah fungsi Triplet Loss. Triplet Loss digunakan untuk membantu model menerapkan margin antara wajah dengan identitas yang berbeda.



Gambar 2.3 Struktur Model FaceNet (Schroff, Kalenichenko, & Philbin, 2015)

Gambar 2.2 menunjukkan struktur model FaceNet yang terdiri atas sekelompok lapisan input dan deep CNN diikuti dengan normalisasi L_2 yang menghasilkan *face embedding*. Kemudian dilanjutkan dengan *triplet loss* ketika melatih.



Gambar 2.4 Triplet Loss (Schroff, Kalenichenko, & Philbin, 2015)

Gambar 2.4 menunjukkan *Triplet Loss* yang akan meminimalisir jarak antara suatu *anchor* dan *positive* dari identitas yang sama, dan akan memperbesar jarak antara *anchor* dan *negative* dari identitas yang berbeda.

Suatu *embedding* direpresentasikan dengan $f(x) \in R^d$ yang menyimpan suatu gambar x pada *d-dimensional Euclidean space*. *Embedding* ini diharuskan untuk berada di *d-dimensional hypersphere* $||f(x)||_2 = 1$. *Loss* ini didukung dari konteks *nearest-neighbor classification*. Disini ingin dipastikan bahwa gambar x_i^a (*anchor*) milik seseorang lebih dekat pada seluruh gambar x_i^p (*positive*) dari orang yang sama dibandingkan dengan gambar x_i^n (*negative*) dari orang yang berbeda. Oleh karena itu diinginkan sebagai berikut :

$$\left|\left|f(x_i^a) - f\left(x_i^p\right)\right|\right|_2^2 + a < \left|\left|f(x_i^a) - f(x_i^n)\right|\right|_2^2,$$
$$\forall \left(f(x_i^a), f\left(x_i^p\right), f(x_i^n)\right) \in T \qquad (2\text{-}1)$$

Di mana a merupakan batas yang dipaksakan antara pasangan *positive* dan *negative*. T merupakan himpunan dari seluruh kemungkinan *triplets* dalam *training set* dan memiliki kardinalitas N. *Loss* dihitung dengan menjumlahkan hasil dari seluruh selisih *embedding* $f(x)$ ditambah dengan margin atau α. Di mana $f(x_i^a)$ merupakan *embedding* untuk *anchor*, $f(x_i^p)$ merupakan *embedding* untuk gambar wajah orang yang sama, dan $f(x_i^n)$ merupakan *embedding* untuk gambar wajah orang yang berbeda. *Loss* dihitung dengan rumus sebagai berikut :

$$L = \sum_i^N [\left|\left|f(x_i^a) - f(x_i^p)\right|\right|_2^2 - \left||f(x_i^a) - f(x_i^n)\right||_2^2 + a] \quad (2\text{-}2)$$

Gambar 2.4 menunjukkan hasil setelah melalui proses FaceNet, hasil tersebut merepresentasikan fitur dari gambar wajah seseorang. Fitur ini kemudian dapat dibandingkan dengan fitur gambar wajah lainnya dengan menghitung ukuran kesamaannya. Salah satu teknik yang dapat digunakan adalah *cosine similarity*.

```
[[-4.86684311e-03  3.57899517e-02  1.51629150e-02  1.60773823e-04
  -7.22369552e-02  7.95743689e-02 -6.26536757e-02  2.38503516e-03
   1.17516648e-02 -7.78794754e-03 -2.89064422e-02  8.81903339e-03
  -3.61743034e-03  3.06959078e-02 -1.01552601e-03 -3.32860462e-02
  -3.37014906e-02 -1.45826284e-02 -6.49634283e-04  2.54215929e-03
  -1.57942027e-01  7.88515285e-02 -5.06995767e-02  1.57651287e-02
  -9.43671260e-03  1.06244525e-02 -5.70593998e-02  7.04967603e-02
   1.64238647e-01 -2.11258605e-02 -1.14261024e-02  2.76883930e-01
   4.55173664e-02  3.15439189e-03 -2.02442724e-02  9.76462960e-02
  -2.99152522e-03 -2.01076232e-02  4.39284416e-03 -8.28727409e-02
   1.35153672e-02  6.54782634e-03  1.94585919e-02 -1.42111266e-02
   9.49807744e-03 -4.32563014e-02 -8.00660923e-02  1.62734333e-02
  -1.42509993e-02  5.93235530e-02  1.13688512e-02 -6.88219583e-03
  -2.55074501e-01 -3.37440521e-03 -3.40861902e-02  1.11247953e-02
   9.35816914e-02  5.47178183e-03 -1.22333772e-01  3.48285250e-02
   5.11324853e-02 -1.08532801e-01 -9.01615471e-02 -3.75422090e-02
  -1.96674746e-02  6.92519695e-02 -1.01281554e-02  6.59857458e-03
   9.63362958e-03  3.08229518e-03 -4.14677970e-02 -1.59562137e-02
  -7.69904442e-03  1.46370130e-02 -6.21918999e-02  4.67659952e-03
   2.31044320e-03 -5.09361154e-04  1.92156643e-01  1.68968663e-02
  -1.36827538e-02  2.58799233e-02 -1.82680991e-02  2.68259794e-01
  -1.20693311e-01 -2.70730630e-03 -5.79523947e-03  3.41457203e-02
   1.72905512e-02 -1.71060175e-01  9.46614221e-02 -5.30435238e-03
   1.38614764e-02 -3.11830677e-02  6.28781551e-03 -1.23911187e-01
  -3.11936457e-02  6.17568716e-02  6.30548166e-04 -5.48692932e-03
   3.96855315e-03 -1.97957400e-02 -6.17158646e-03  1.49552256e-03
  -4.98866709e-03  8.05946533e-03 -1.65658221e-01  1.94264867e-03
  -1.78184104e-03  1.13912821e-02 -1.14198759e-01  1.03200600e-02
   1.16948728e-02  2.60102928e-01  1.26949726e-02  1.67274594e-01
  -3.66585725e-03 -4.52584550e-02  5.62946573e-02  1.40183672e-01
   2.68469065e-01 -7.73321372e-03 -1.56558141e-01 -3.14313034e-03
   1.21433614e-03 -3.44745023e-03  7.74152484e-03  1.34264352e-03
   2.03550328e-03 -5.09233810e-02  1.39346635e-02  2.80557871e-02
  -3.70415370e-03 -5.70967533e-02  6.91141337e-02 -1.75000578e-02
  -2.01833561e-01  1.48790339e-02  8.26006755e-03  1.71971265e-02
  -7.62380334e-03  9.90328845e-04 -2.38569966e-03  1.53073687e-02
  -1.36289343e-01  8.32647160e-02 -2.90480219e-02 -1.50971301e-03
   1.30020780e-02 -9.86611377e-03 -1.11631779e-02 -8.23850781e-02
  -1.25772217e-02 -3.48512419e-02 -8.80226586e-03 -2.39150855e-03
   3.56483762e-03  8.28411710e-03 -9.53763276e-02 -3.45793692e-03
  -8.43069702e-02 -2.79903034e-04 -1.56892836e-02  9.78851458e-04
   6.62902882e-03  2.68163458e-02  2.42406242e-02 -8.34838599e-02
  -3.21833929e-03  1.00883667e-03  1.43289983e-01  1.53715953e-01
  -3.51880398e-03 -2.86211614e-02  1.78163033e-02  7.59298354e-03
  -5.24892211e-02 -1.01227105e-01 -1.71121024e-02  1.19623411e-02
  -6.80727959e-02  1.19247800e-02 -1.85849075e-03 -1.57253526e-03
   1.30434856e-01  5.52805932e-03 -1.40591651e-01  5.47555424e-02
   1.49417361e-02 -4.72348407e-02 -8.97105411e-02 -1.15640284e-02]]
```

Gambar 2.5 Hasil Proses FaceNet

2.7 Cosine Similarity

Cosine similarity merupakan salah satu metode perhitungan kemiripan dalam *data mining*. *Cosine similarity* adalah pengukuran kemiripan antara dua objek data. *Cosine similarity* terdiri dari kata *cosine* yang merupakan sudut antara dua besaran yang memiliki nilai berdimensi, dan *similarity* yang berarti kemiripan (Atmaja, 2022). Dalam *cosine similarity*, objek data dianggap sebagai vektor.

$$Sim(A,B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A \cdot B}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\sqrt{\sum_{i=1}^{n}(B_i)^2}} \qquad (2\text{-}3)$$

Rumus 2-3 menunjukkan rumus cosine similarity, di mana Sim(A,B) merupakan kemiripan, A dan B adalah nilai besaran yang ingin dibandingkan, i adalah perulangan, dan n adalah jumlah nilai besaran.

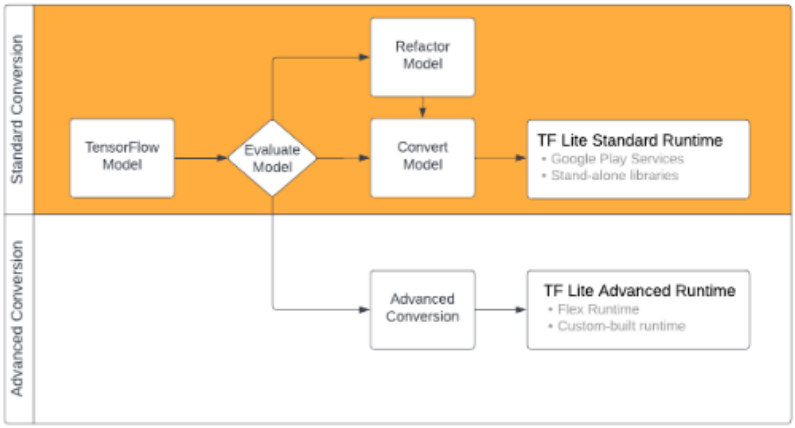
2.8 Tensor Flow Lite

TensorFlow dikembangkan oleh tim Google, di mana framework TensorFlow berifat open source sehingga dapat digunakan secara gratis bagi semua orang. TensorFlow digunakan untuk pengembangan machine learning. TensorFlow dapat digunakan pada banyak bahasa, Python, R, Swift, Java, Kotlin, Java Script, dan lainnya.

TensorFlow Lite adalah *extension* dari TensorFlow. TensorFlow Lite digunakan untuk melakukan konversi dan juga optimisasi model bagi perangkat *mobile*. TensorFlow Lite merupakan framework yang open-source dan cross-platform yang mengubah model pre-trained di TensorFlow menjadi format khusus yang mengoptimasi kecepatan dan penyimpanannya. Model ini kemudian dapat digunakan pada perangkat mobile dengan Android, iOS, atau perangkat berbasis Linux.

Gambar 2.7 Menunjukkan proses konversi model TensorFlow menjadi model TensorFlow Lite. Hal yang pertama dilakukan adalah mengevaluasi model untuk menentukan apakah model tersebut dapat dikonversi langsung. Evaluasi ini akan menentukan apakah konten dari model tersebut didukung oleh runtime environment standar TensorFlow Lite berdasarkan operasi TensorFlow yang digunakan. Jika

model tersebut menggunakan operasi diluar dari yang didukung, maka model tersebut dapat di-refactor atau menggunakan teknik konversi yang lebih canggih.



Gambar 2.7 Proses Konversi TensorFlow Lite (TensorFlow, 2023)

2.9 Java

Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh James Gosling pada awal tahun 1990-an. Tujuan dari proyek ini adalah untuk mengembangkan perangkat digital seperti televisi. Namun karena C++ membutuhkan banyak memori, Gosling mencari alternatif lain dan dikembangkanlah bahasa pemrograman Java. Sampai saat ini, bahasa pemrograman Java telah dipakai dalam pengembangan bagian back-end dari software, website, dan juga aplikasi Android (Aprilia, 2021).

Java adalah salah satu bahasa pemrograman paling terkenal dan banyak digunakan di dunia. Meskipun merupakan bahasa pemrograman berorientasi objek, Java tetap mendukung tipe data primitif seperti int, char, dan lainnya. Sintaks yang dimiliki oleh Java mirip dengan C/C++, hanya saja Java tidak mendukung fungsionalitas pemrograman tingkat rendah dan kodenya selalu ditulis dalam bentuk objek dan kelas.

2.10 Hive

Hive merupakan database yang ringan dan cross-platform, di mana Hive dapat berjalan di mobile, desktop, maupun web. Hive ditulis dengan bahasa Dart, hal ini memberikan keunggulan dibandingkan SQFLite yang tidak mendukung

berjalan di web, sedangkan Hive tidak memiliki native dependency sehingga dapat berjalan di web.

Hive menggunakan konsep *boxes* untuk menyimpan data di database. Suatu *box* mirip dengan tabel di basis data SQL, tetapi *box* lebih fleksibel dan hanya dapat menangani relasi antar data yang sederhana. Sebelum mengakses data yang tersimpan dalam suatu *box*, pertama *box* tersebut harus dibuka terlebih dahulu. Hal ini akan memuat seluruh data yang tersimpan dalam *box* di penyimpanan lokal ke memori, sehingga data yang tersimpan dalam suatu *box* dapat diakses dengan mudah.

2.11 Database

Database atau basis data adalah kumpulan data yang terorganisir atau terstruktur. Basis data dapat diakses dan disimpan pada sistem komputer. Untuk mengelola basis data, dibutuhkan prangka lunak Database Management System (DBMS). Dalam basis data, data akan diatur dalam tabel yang memiliki baris dan kolom. Data akan diberi indeks agat dapat diperbarui, dihapus, atau diperluas dengan mudah. Berikut adalah jenis-jenis basis data :

a. *Relational Database*, merupakan basis data yang terdiri dari sekumpulan tabel dengan data berkategori yang sudah ditentukan sebelumnya.
b. *Distributed Database*, merupakan basis data dengan sebagiannya disimpan pada lokasi fisik yang berbeda, di mana prosesnya tersebar pada bagian-bagian yang berbeda dalam jaringan.
c. *Cloud Database*, merupakan basis data yang berjalan pada platform *cloud computing*. Layanan basis data akan memberikan akses ke basis data tersebut.

2.12 Postgre SQL

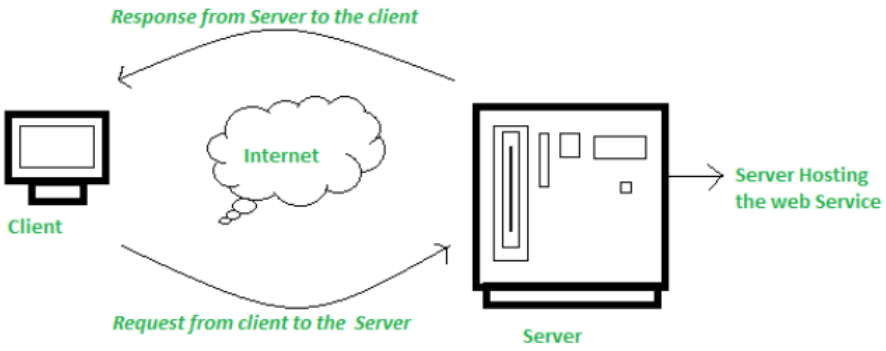
PostgreSQL atau Postgres adalah *Object Relational Database Management System* (ORDBMS) (Praba & Safitri, 2020). PostgreSQL dikembangkan oleh Michael Stonebraker dari Universitas California di Berkley. Tujuan dari proyek Postgres adalah untuk menambahkan fitur seperti kemampuan untuk mendeifinisikan berbagai tipe data dan sepenuhnya mendeskripsikan relasi atau

hubungan, sesuatu yang digunakan secara luas tetapi sepenuhnya dikelola oleh pengguna.

PostgreSQL memiliki sifat *open source* yang berarti dapat digunakan oleh siapa saja secara bebas. Kemampuan *Structured Query Language* (SQL) untuk melakukan *transactions*, *subqueries*, *triggers*, dan lainnya didukung oleh PostgreSQL. Beberapa keunggulan Postgres yaitu tipe yang ditentukan pengguna, tabel warisan, mekanisme penguncian, *nested transaction*, pemulihan tepat waktu, juga MVCC (*Multi-version concurrency control*).

2.13 Web Service

Web Service atau layanan web merupakan suatu sistem perangkat lunak dengan kemampuan untuk mendukung interaksi dan interoperabilitas antar sistem dengan sistem yang lain di dalam suatu jaringan. *Web service* memberikan fasilitas yang menyediakan layanan informasi atau data kepada sistem lain agar kedua sistem ini dapat saling berinteraksi melaui layanan tersebut. Sistem atau aplikasi yang berbeda ini akan menggunakan seperangkat protokol pada *web service* untuk melakukan pertukaran data. *Web service* dapat digunakan oleh program dengan berbagai bahasa pemrograman dan berjalan di berbagai platform untuk saling bertukat data melaui jaringan seperti internet. Layanan ini akan mengirimkan data dengan format JSON atau XML, dengan format ini, data dapat diakses oleh sistem yang berbeda sistem operasi, bahasa pemrograman, maupun platform.



Gambar 2.8 Cara kerja *Web Service* (GeeksforGeeks, 2021)

Gambar 2.8 menggambarkan cara kerja *web service* dengan sederhana. Pertama klien akan mengirimkan permintaan kepada server yang akan menjadi *host* dari *web service* tersebut. Permintaan ini dikirimkan menggunakan prosedur jarak jauh. Panggilan kepada metode yang ada pada *web service* dikenal sebagai Remote Procedure Calls (RPC). Data yang ditukarkan antara klien dan server merupakan data XML (*Extensible markup language*). Setelah memproses permintaan dari klien, server akan mengirimkan respon dengan menggunakan HTTP standar dalam bentuk SOAP (*Simple Object Access Protocol*).

2.14 REST API

Representational State Transfer Application Programming Interface atau REST API merupakan arsitektur dari API yang menggunakan *Hypertext Transfer Protocol* (HTTP) sebagai media untuk pertukaran data. Data yang ada dalam REST API berbentuk *Javascript Object Notation* (JSON) yang memiliki mobilitas data lebih cepat.

Berikut adalah contoh dari metode HTTP yang biasa digunakan dalam REST API :

a. GET, metode ini biasa digunakan untuk mendapatkan data atau informasi yang ada.
b. POST, metode ini biasa digunakan untuk memasukkan data.
c. PUT, metode ini biasa digunakan untuk pembaruan data yang ada.
d. DELETE, metode ini biasa digunakan untuk menghapus data.

2.15 Visual Studio Code

Visual Studio Code merupakan *software code editor* yang dapat dijalankan di berbagai *operating system* seperti Windows, Linux, dan juga MacOS. Visual Studio Code dikembangkan oleh Microsoft pada tahun 2015. Software ini merupakan *code editor* yang *opensource* atau gratis untuk digunakan siapapun. Visual Studio Code memiliki beragam *extension*. Dengan adanya berbagai macam *extension* memungkinkan Visual Studio Code untuk mengetik atau mengubah berbagai bahasa pemrograman mulai dari JavaScript, TypeScript, Dart, Kotlin, dan lainnya.

Hal ini juga memungkinkannya untuk menjalankan berbagai runtime environment seperti PHP, Python, Java, maupun .NET.

Berikut adalah beberapa fitur penting yang dimiliki Visual Studio Code :

a. Github *Integration*, Github sendiri perupakan platform project management yang memungkinkan seseorang untuk saling berkolaborasi dan berbagi kode dengan tim nya. Dengan adanya integrasi Github, Visual Studio Code data menjalankan perintah-perintah seperti *pull*, *push*, *commit*, maupun *merge* secara langsung.
b. *Basic Editing*, fungsi utama Visual Studio Code sebagai code editor adalah untuk menuliskan kode. Visual Studio Code memiliki fitur untuk *formatting code*, *auto-save*, *hotkey*, *shortcut*, dan berbagai fitur pendukung untuk menuliskan kode.
c. *Extension Marketplace*, dengan memiliki *extension marketplace*, pengguna dapat memasang berbagai alat atau ekstensi diluar Visual Studio Code.
d. *Debugging*, fitur ini memudahkan pengguna untuk mencari kesalahan dalam kode yang ditulis, dengan adanya pesan error yang menunjukkan kesalahannya.
e. *IntelliSense*, fitur *intellisense* memungkinkan Visual Studio Code untuk memberikan saran kemungkinan kode yang ditulis berdasarkan bahasa yang digunakan.

Dengan adanya berbagai fitur tersebut Visual Studio Code menjadi salah satu *code editor* yang paling unggul. Berikut adalah beberapa kelebihan dari Visual Studio Code :

a. Ringan, meskipun memiliki berbagai macam fitur, Visual Studio Code merupakan *code editor* yang cukup ringan. Hal ini karena dengan adanya *extension marketplace* memungkinkan pengguna untuk mengunduh fitur yang dibutuhkan saja.
b. *Multiplatform*, seperti yang telah disebutkan di awal, Visual Studio Code yang dikembangkan oleh Microsoft ini kompatibel dengan berbagai *operating system*. Beberapa contohnya adalah Linux, Windows, MacOS, dan lainnya.

c. Mendukung berbagai bahasa pemrograman, fitur *extension marketplace* memungkinkan pengguna untuk mengunduh *extension* untuk berbagai bahasa pemrograman yang diinginkan.
d. Fitur yang lengkap dan gratis, *extension marketplace* yang ada pada Visual Studio Code menyajikan berbagai *extension* untuk mendukung penulisan kode pengguna dan dapat diakses secara gratis.

2.16 Netbeans

Netbeans merupakan aplikasi *Integrated Development Environment* atau IDE yang berbasis Java. IDE merupakan sistem pengembangan yang diintegrasikan pada suatu perangkat lunak. Netbeans dijalankan diatas Swing, teknologi yang membuat agar aplikasi pengembangan dapat dijalankan di berbagai platform, seperti Windows, Linux, MacOS, juga Solaris. Netbeans merupakan aplikasi *opensource*, yang berarti dapat diakses oleh siapapun secara gratis. Netbeans dikembangkan sejak tahun 1996 dengan nama Xelfi, proyek Java IDE yang dikerjakan oleh sekelompok mahasiswa dari Charles University, Prague. Kemudian pada tahun 1999 dibeli oleh Sun Mycrosystem untuk dikembangkan lebih lanjut menjadi aplikasi yang sekarang ini.

Fungsi utama Netbeans adalah media untuk menuliskan kode, *compile*, juga *debugging*. Netbeans mendukung berbagai bahasa pemrograman seperti Java, C/C++, PHP, JavaScript, Groovy, dan juga Ruby. Dengan Netbeans yang mendukung berbagai bahasa pemrograman memungkinkan Netbeans untuk menghasilkan program yang berjalan di *web, mobile, enterprise,* dan juga *desktop*.

2.17 Penelitian Terdahulu

Penelitian mengenai pengenalan wajah menggunakan facenet yang sebelumnya dilakukan oleh (Evelyn, Adipranata, & Gunadi, 2022). Penelitian ini dilakukan perbandingan antara dua rumus perhitungan, rumus L2Norm dan rumus *cosine similarity*. Penelitian ini dilakukan dengan cara membandingkan data latih dengan dua kategori data uji dengan *threshold* yang berbeda-beda. Data uji yang pertama berisikan data wajah yang berbeda dengan data latih, sedangkan data uji yang kedua berisikan data wajah yang sama dengan data latih. Dari hasil

penelitiannya diperoleh bahwa rumus L2Norm paling optimal dengan menggunakan *threshold* 8.0f dengan nilai terbaik 5.8973804 dan rumus cosine similarity paling optimal dengan menggunakan threshold 0.5f dengan nilai terbaik 0.5104218.

Penelitian lainnya terkait dengan aplikasi untuk absensi dilakukan oleh (Basurah, Swastika, & Kelana, 2019). Pada penelitian ini dikembangkan sebuah aplikasi berbasis web untuk melakukan absensi. Pada aplikasi yang dikembangkan, wajah akan diproses melalui tiga tahap, *face detection*, *liveness detection*, dan *face recognition*. *Liveness detection* dilakukan dengan meminta pengguna untuk melakukan beberapa ekspresi acak sebanyak tiga kali. Apabila pengguna berhasil mendapatkan tiga skor, maka dianggap tidak melakukan *spoofing*. Untuk mendeteksi ekspresi dan mengenali wajah pengguna, penulis menggunakan model faceapi.js. Selain itu juga dilakukan *object detection* untuk menghindari pengguna yang melakukan kecurangan melalui video call. Dari penelitian ini diperoleh akurasi untuk pengenalan wajah sebesar 85%, akurasi untuk mendeteksi ekspresi pengguna sebesar 82.5%, dan akurasi untuk mendeteksi objek sebesar 100%.

Penelitian mengenai aplikasi absensi dengan pengenalan wajah menggunakan metode *Viola Jones* dan algoritma *Local Binary Pattern Histogram* (LBPH) dilakukan oleh (Buana, 2021). Metode Viola Jones digunakan untuk mendeteksi objek wajah, dan algoritma LBPH digunakan untuk mengenali wajah tersebut. Data training yang disimpan sebanyak 100 gambar, data *training* akan dibandingkan dengan gambar *real-time* yang tertangkap dari webcam. Pengujian pada aplikasi ini dilakukan dari berbagai jarak dan sudut kemiringan wajah. Jarak yang diuji ada 30 cm, 40 cm, 60 cm, 100 cm, 170 cm, dan 200 cm. Sedangkan sudut kemiringan wajah yang diuji ada tegak lurus, 10° ke kanan, 20° ke kanan, 30° ke kanan, 10° ke atas, 20° ke atas, dan 30° ke atas. Dari pengujian yang dilakukan dapat disimpulkan bahwa pada jarak 30 cm dan 200 cm dari kamera, wajah tidak dapat terdeteksi. Juga tingkat kemiringan wajah tegak lurus hingga sekitar 20° kemiringan masih bisa dikenali, sedangkan 30° ke atas maupun ke kanan sudah tidak dapat dikenali.

Penelitian pengenalan wajah dengan *machine learning* menggunakan metode *K-Nearest Neighbor* dilakukan oleh (Iskandar, Abdurahman, & Abdurahman, 2022). Penelitian memiliki dataset yang terdiri atas 255 gambar wajah dari lima

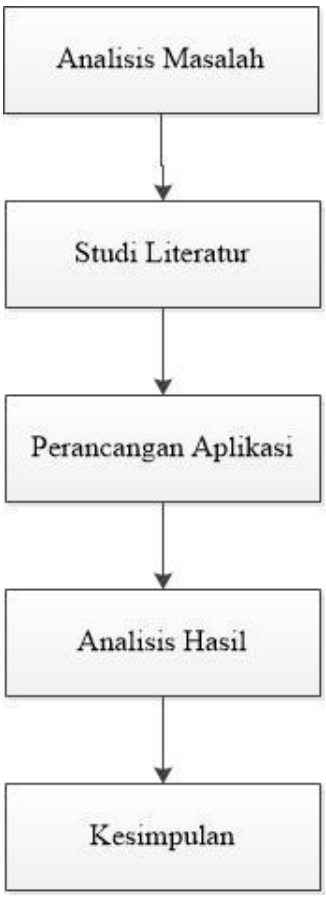
orang siswa, di mana masing-masing siswa memiliki 51 gambar wajah dengan posisi yang berbeda-beda. Dari penelitian ini didapati bahwa metode K-NN mendapatkan hasil pengenalan wajah sesuai dengan confusion matrik dengan nilai akurasi 94.99%, presisi 94.41%, recall 94.20%, dan k-fold F1 94.19.

Penelitian mengenai pengenalan wajah menggunakan metode *eigenface* dilakukan oleh (Putra, Fitri, & Fitri, 2021). Algoritma eigenface merupakan algoritma image processing yang mengangkat konsep Principal Component Analysis (PCA). Algoritma ini berutujuan untuk mengurangi dimensionalitas juga mencari nilai vektor paling tinggi. Data training merupakan 10 gambar wajah yang diambil dengan jarak kurang lebih 50 cm dari kamera. Pengujian dilakukan sebanyak tiga kali dengan tiga orang yang berbeda. Hasil dari pengujian didapatkan bahwa akurasi dari pengujian pertama sebesar 83%, akurasi dari pengujian kedua sebesar 65%, dan akurasi pada pengujian ketiga sebesar 65%.

BAB III
METODE PENELITIAN

3.1 Alur Penelitian

Proyek Tugas Akhir ini memiliki tujuan untuk mengembangkan aplikasi absensi bagi karyawan yang mempermudah pendataan absen menggunakan framework Flutter agar aplikasi dapat digunakan pada berbagai platform. Proses pengerjaan dari Tugas Akhir dibagi menjadi beberapa tahap. Berikut alur tahap pengerjaan Tugas Akhir.



Gambar 3.1 Alur Tugas Akhir

Gambar 3.1 menunjukkan tahap pengerjaan Tugas Akhir yang dimulai dengan analisis masalah. Dalam tahap analisis masalah hal yang dilakukan adalah identifikasi masalah dan memberikan batasan pada masalah tersebut sebagai tahap

awal dalam pengerjaan Tugas Akhir ini. Kemudian dilanjukan dengan studi literatur untuk mempelajari pengembangan aplikasi serupa dan memahami kebutuhan untuk membangun aplikasi tersebut. Selain itu juga mempelajari dasar-dasar teori yang berkaitan dengan Tugas Akhir ini baik bersumber dari buku maupun e-book dan jurnal. Tahap selanjutnya adalah perancangan aplikasi. Pada tahap ini dirancang alur, basis data, dan tampilan dari aplikasi tersebut. Selanjutnya adalah analisis hasil, yaitu dilakukan pengujian terharap fitur dari aplikasi tersebut serta survei terkait dengan fitur dan tampilan aplikasi. Pada akhirnya dilakukan analisis terhadap hasil dari pengujian fitur dan juga survei untuk pengambilan kesimpulan dan saran untuk pengembangan aplikasi serupa kedepannya.

3.2 Analisis Masalah

Analisis masalah yang ada yaitu dengan jumlah karyawan yang besar dan fleksibilitas untuk kerja dari rumah maupun dari kantor pada masa pandemi membuat pendataan absensi menjadi sulit. Selain itu juga terdapat masalah di mana penggunaan sistem operasi selain Android semakin berkembang. Pada proyek Tugas Akhir ini akan dirancang aplikasi absensi menggunakan framework Flutter dan sistem pengenalan wajah FaceNet untuk mempermudah proses absensi dengan aplikasi yang dapat digunakan di berbagai platform.

Pada aplikasi ini dibutuhkan perangkat keras dan lunak untuk proses pengembangannya. Perangkat keras yang digunakan adalah laptop dengan spesifikasi processor Intel ® Core ™ i7-11370H, RAM 16 GB, dan 64 bit *operating system*. Perangkat keras lainnya adalah telepon genggam untuk menjalankan aplikasi yang akan dikembangkan. Perangkat lunak yang dibutuhkan adalah Visual Studio Code untuk menuliskan kode Dart dengan framework Flutter dan NetBeans untuk menuliskan kode *webservice* dengan Java.

3.3 Studi Literatur

Pada tahap ini dilakukan studi literatur terkait dengan metode-metode pengenalan wajah dan *framework* Flutter yang menjadi acuan dalam pengembangan aplikasi ini. Proses ini bertujuan untuk mempelajari proses dari pengenalan wajah dan penerapannya pada aplikasi *mobile*.

3.4 Perancangan Aplikasi

Aplikasi akan melakukan pengecekan lokasi pengguna untuk memastikan pengguna berada di rumah atau kantor sesuai dengan jadwal kerjanya. Kemudian akan dilakukan pengenalan wajah. Pengenalan wajah pengguna pada aplikasi ini akan menggunakan gabungan dari Firebase Vision Face yang merupakan bagian dari ML Kit untuk mendapatkan area wajah pengguna dan FaceNet untuk mengenali wajah tersebut. Setelah absen, pengguna dapat melihat riwayat absennya.

Perancangan berikutnya adalah perancangan use case diagram. Use case diagram digunakan untuk menentukan fitur yang dapat diakses dan dugunakan oleh pengguna. Dari use case ini akan menentukan fitur apa saja yang akan dikembangkan pada aplikasi.

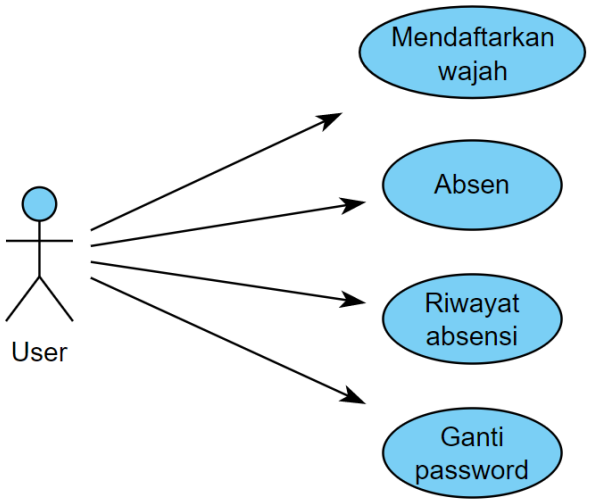
Tahap perancangan berikutnya adalah perancangan basis data. Pada tahap ini basis data dirancang dengan Entity Relationship Diagram atau ERD. ERD sendiri terbagi menjadi Conceptual Data Model (CDM) dan Physical Data Model (PDM). Perancangan basis data ini akan menentukan relasi antar tabel juga struktur data berdasarkan kebutuhan alur dari datanya.

Hal yang perlu dilakukan berikutnya adalah perancangan *mock up*. *Mock up* dirancang untuk memberikan *interface* dan *experience* yang baik bagi pengguna. *Mock up* akan menentukan tata letak dari komponen-komponen pada suatu halaman aplikasi dan membantu pembuatan *layout* aplikasi.

3.4.1 *Use Case* Diagram

Use case merupakan interaksi atau hubungan antara aktor dengan sistem. Gambar 3.2 menunjukkan *use case diagram* untuk Tugas Akhir ini. *Use case diagram* digunakan untuk menunjukkan fitur yang dapat diakses oleh suatu entitas. Dapat dilihat dalam *use case diagram* ini terdapat satu entitas yang disebut sebagai *user*. *User* disini dapat mengakses tiga fitur yang ada pada proyek Tugas Akhir ini. Fitur pertama adalah mendaftarkan wajah, pada fitur ini *user* yang belum mendaftarkan wajahnya dapat mengambil foto yang kemudian akan diproses dan disimpan pada aplikasi. Setelah mendaftarkan wajah, *user* dapat menggunakan fitur

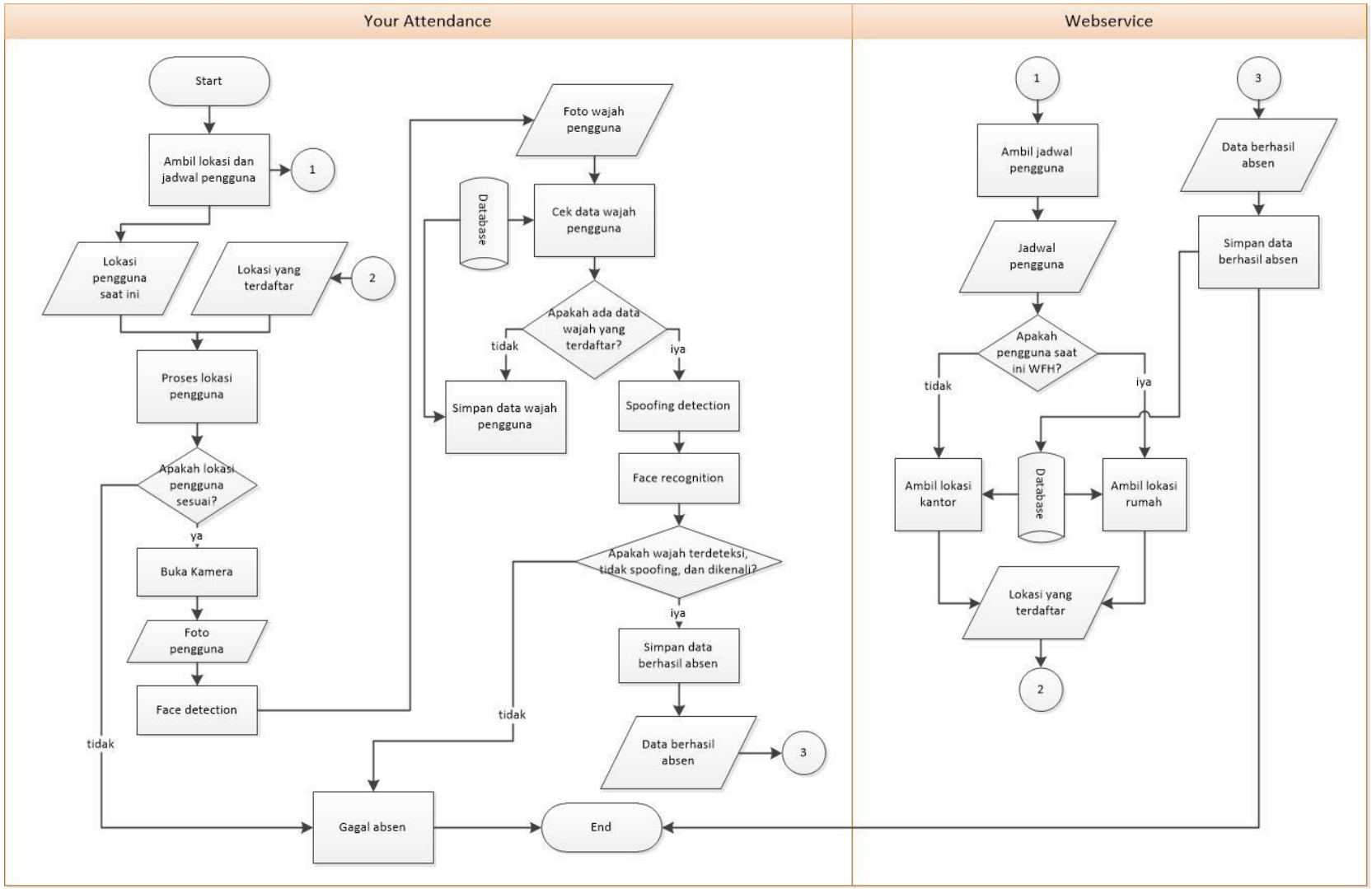
absen, di mana *user* dapat melakukan absensi untuk pendataan kehadiran dengan mengambil foto wajahnya yang kemudian akan dilalui proses *face recognition* untuk memastikan bahwa wajah dalam foto merupakan wajah *user* sesuai dengan data yang telah disimpan ketika mendaftarkan wajah. Kemudian setelah melakukan absen *user* dapat melihat riwayat absennya dengan fitur riwayat absensi. Selain itu apabila *user* melupakan *password* yang dimiliki maka *user* dapat menggunakan fitur ganti password untuk mengubah *password*-nya.



Gambar 3.2 Use Case Diagram

3.4.2 Diagram Flowchart

Tahap berikutnya adalah perancangan diagram flowchart. Diagram ini dirancang untuk menentukan alur dari fitur yang akan dikembangkan untuk perangkat dengan sistem operasi Android dan iOS. Berdasarkan use case diagram yang telah dirancang sebelumnya, terdapat empat fitur yang akan dikembangkan pada proyek Tugas Akhir ini, alur untuk mendaftarkan wajah, alur untuk absensi, alur untuk melihat riwayat absensi, dan alur untuk mengganti *password*. Pada perancangan diagram flowchart ini akan dibuat tiga diagram, diagram flowchart untuk alur mendaftarkan wajah serta absensi, diagram untuk alur pada melihat riwayat absensi, serta diagram untuk alur pada mengganti *password*.

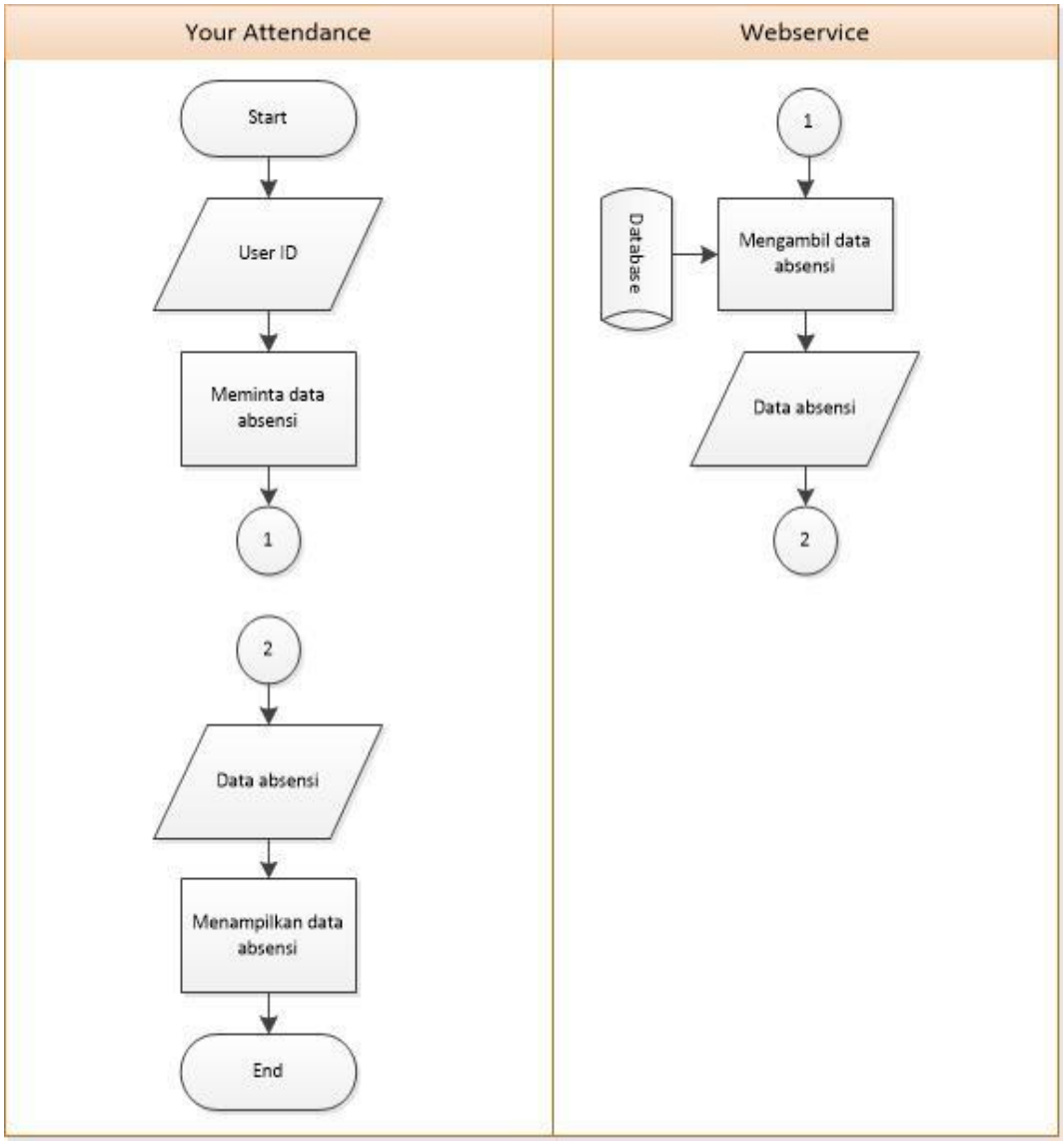


Gambar 3.3 *Flowchart Diagram* Daftar Wajah dan Absensi

Gambar 3.3 menunjukkan alur dari fitur mendaftarkan wajah dan absen. Pertama aplikasi akan mengirimkan request kepada *webservice* untuk memperoleh lokasi di mana pengguna seharusnya berada. *Webservice* akan mengecek jadwal dari pengguna apakah seharusnya saat ini *work from office* atau WFO atau *work from home* atau WFH. Apabila menurut jadwal pengguna WFO, maka alamat yang diambil adalah alamat kantor. Namun apabila jadwal WFH, maka alamat yang diambil adalah alamat rumah yang sebelumnya sudah didaftarkan. Alamat tersebut kemudian akan dibandingkan dengan lokasi pengguna saat ini.

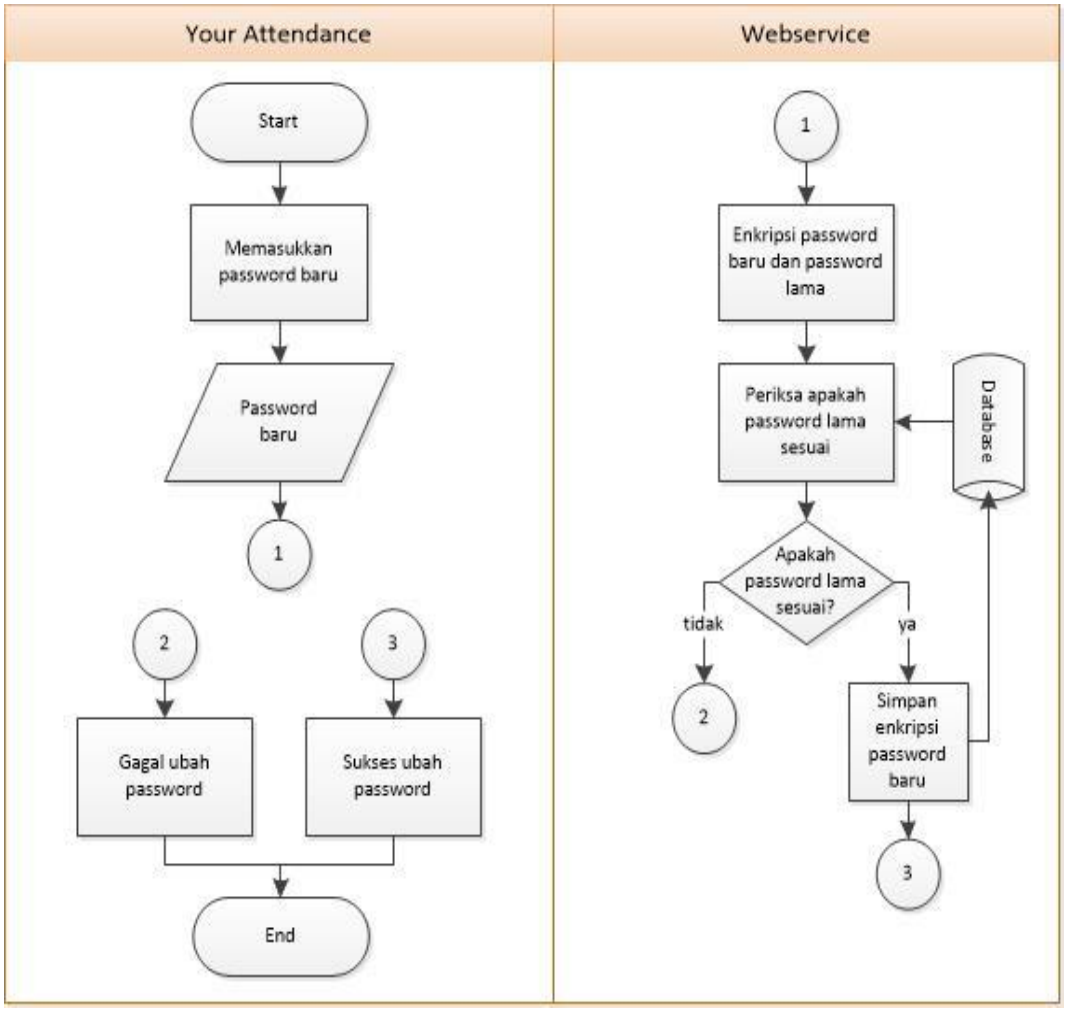
Apabila jarak antara pengguna dengan lokasi tempat pengguna seharusnya berada, baik itu di kantor atau di rumah, melebihi satu kilometer, maka lokasi pengguna dianggap tidak valid dan proses absensi gagal. Jika jaraknya kurang dari satu kilometer dari tempat pengguna seharusnya berada, maka lokasi pengguna sesuai dan dilanjutkan pada proses berikutnya.

Tahap selanjutnya akan dilakukan *face detection* untuk menentukan area dari wajah pengguna. Setelah berhasil ditemukan adanya wajah, maka akan diperiksa adanya data wajah yang tersimpan pada aplikasi. Jika belum ada maka wajah tadi akan diproses kemudian disimpan pada aplikasi untuk mendaftarkan wajah tersebut. Wajah yang disimpan ini akan menjadi *sample* untuk dibandingkan ketika proses presensi kedepannya. Tetapi apabila ditemukan data wajah yang sudah tersimpan atau terdapat satu *sample* wajah pada aplikasi, maka dilanjutkan dengan proses *spoofing detection*. Pada proses ini pengguna akan diminta untuk melakukan gerakan sesuai dengan instruksi yang diberikan. Instruksi yang diberikan dapat berupa menghadap ke kiri, kanan, atas, maupun bawah. Setelah berhasil melalui proses *spoofing detection*, wajah yang diambil diawal akan melalui proses *face recognition* di mana wajah tersebut akan dibandingkan dengan data *sample* wajah yang terdaftar pada aplikasi. Proses pengenalan wajah dilakukan pada bagian *front end* pada aplikasi. Bila sistem tidak dapat mengenali wajah tersebut, maka proses absensi gagal. Namun bila wajahnya terdeteksi atau sama dengan wajah yang sudah terdaftar pada aplikasi, maka data presensi yang berhasil berupa waktu dan tempat presensi akan dikirimkan ke *webservice* untuk disimpan di *database*.



Gambar 3.4 *Flowchart Diagram* Riwayat Absen

Gambar 3.4 menunjukkan alur dari fitur untuk melihat riwayat absensi. Aplikasi akan mengirimkan ID *user* untuk meminta data absensi yang dimiliki oleh *user* tersebut. Kemudian *webservice* akan memperoleh data absensi *user* dari *database*. Data absensi yang didapatkan dari *webservice* akan ditampilkan pada aplikasi *user*.

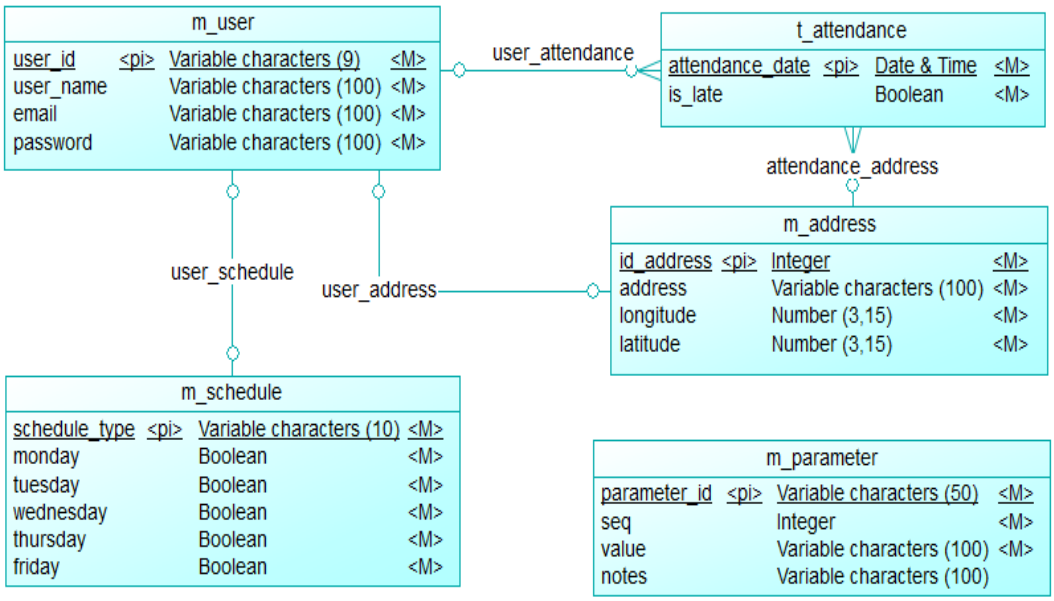


Gambar 3.5 *Flowchart Diagram* Ganti Password

Gambar 3.5 menunjukkan *flowchart diagram* untuk alur ganti *password*. Pertama *user* akan memasukkan *password* baru yang diinginkan. Kemudian *password* tersebut akan dikirmkan pada *webservice*. *Webservice* akan mengenkripsi *password* baru dan lama dengan metode SHA-256. Hasil dari enkripsi *password* lama kemudan akan dibandingkan dengan *password* yang ada pada *database*. Apabila *password* yang lama dengan yang ada pada *database* sama, maka *password* baru akan disimpan dan proses ganti *password* berhasil. Namun bila *password* lama dengan yang ada pada *database* berbeda, maka *password* baru tidak akan disimpan ke *database* dan proses ganti *password* gagal.

3.4.3 Entity Relationship Diagram (ERD)

Tahap berikutnya adalah merancang *Entity Relationship Diagram* (ERD). ERD dirancang untuk membantu menentukan entitas serta atribut yang dibutuhkan dalam alur absensi, baik itu untuk memeriksa jadwal, memeriksa dan menyimpan absensi, mengganti password, dan tentunya untuk absen. Dengan perancangan ini juga dapat membantu untuk menentukan relasi yang dimiliki setiap entitas yang ada. ERD terbagi menjadi dua diagram, *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM).



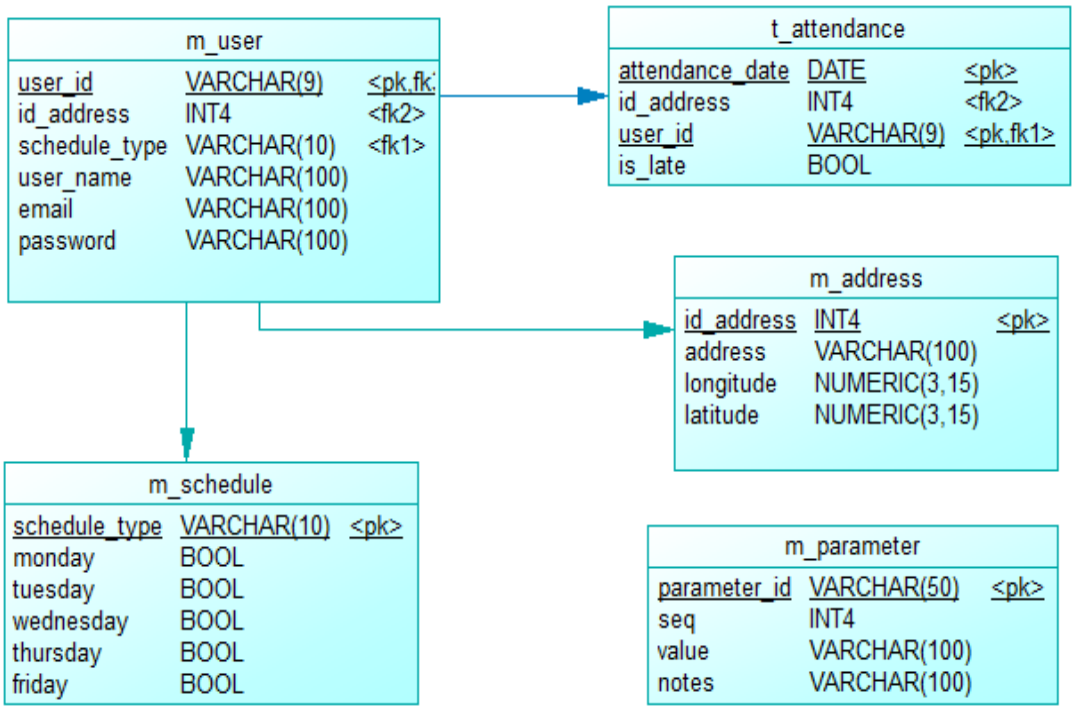
Gambar 3.6 *Conceptual Data Model* (CDM)

CDM merupakan diagram yang berisikan entitas, atribut, relasi, serta kardinalitas suatu basis data. CDM untuk proyek tugas akhir ini dapat dilihat pada gambar 3.6. Pada gambar itu terdapat lima entitas, yaitu m_user, t_attendance, m_address, m_schedule, dan m_parameter.

Tabel m_schedule merupakan tabel yang berisikan data jadwal masuk dalam satu minggu yang dimiliki setiap *shift* dalam perusahaan. Tabel m_user merupakan tabel yang berisikan data karyawan serta user_id dan password yang dibutuhkan untuk proses *login*. Tabel m_address berisikan data alamat termasuk *longitude* dan *latitude* yang dibutuhkan untuk menghitung jarak. Tabel t_attendance merupakan tabel yang berisikan data absensi yang dilakukan karyawan setiap sebelum dan

sesudah jam kerja. Tabel m_parameter berisikan beberapa *value* utama yang dibutuhkan dalam proyek tugas akhir, seperti id_address untuk lokasi kantor, serta jam masuk dan jam pulang kerja.

Tabel m_schedule berelasi *one to one* dengan tabel m_user karena setiap karyawan hanya memiliki satu *shift*. Tabel m_user sendiri berelasi one to one dengan tabel m_address di mana setiap karyawan akan hanya memiliki satu alamat yang terdaftar untuk jadwal *work from home*. Tabel m_user juga berelasi *one to many* dengan tabel t_attendance, di mana setiap karyawan akan melakukan absen setiap harinya. Tabel m_address berelasi one to many dengan tabel t_attendance, di mana setiap alamat dapat digunakan pada banyak data absensi.



Gambar 3.7 *Physical Data Model* (PDM)

PDM merupakan diagram yang mengimplementasikan konsep basis data pada CDM untuk dibentuk menjadi bentuk *physical* dari *database* yang akan dibuat. Dengan adanya PDM akan membantu menentukan *foreign key* yang merelasikan antar tabel.

Gambar 3.7 menunjukkan PDM yang digunakan pada proyek tugas akhir ini. Pada PDM tersebut dapat dilihat pada tabel m_user memiliki *foreign key* schedule_type yang akan menghubungkan tabel m_user dengan tabel m_schedule dan id_address yang akan menghubungkan tabel m_user dengan tabel m_address. Tabel t_attendance memiliki dua *foreign key*, id_address yang menghubungkan tabel t_attendance dengan tabel m_address dan user_id yang menghubungkan tabel t_attendance dengan tabel m_user. Pada diagram ini dapat dilihat bahwa m_parameter tidak memiliki relasi dengan tabel manapun, hal ini karena tabel m_parameter menyimpan data yang tidak memiliki hubungan dengan data pada tabel lainnya.

Tabel 3.1 sampai dengan tabel 3.5 menunjukkan rincian nama kolom, tipe data, serta keterangan atribut dari setiap entitas yang ada pada CDM maupun PDM yang telah dirancang sebelumnya. Mulai dari tabel m_user, tabel m_schedule, tabel m_address, tabel t_attendance, dan tabel m_parameter.

Tabel 3.1 Tabel m_user

Nama Kolom	Tipe Data	Keterangan
user_id	varchar(9)	primary key, not null
schedule_type	varchar(10)	foreign key, not null
id_address	integer	foreign key, not null
user_name	varchar(100)	not null
Email	varchar(100)	not null
password	varchar(100)	not null

Tabel 3.2 Tabel m_schedule

Nama Kolom	Tipe Data	Keterangan
schedule_type	varchar(10)	primary key, not null
Monday	Boolean	not null
Tuesday	Boolean	not null
wednesday	boolean	not null
thursday	boolean	not null
Friday	boolean	not null

Tabel 3.3 Tabel m_address

Nama Kolom	Tipe Data	Keterangan
id_address	integer	primary key, not null
address	varchar(100)	not null
longitude	numeric(3,15)	not null
latitude	numeric(3,15)	not null

Tabel 3.4 Tabel t_attendance

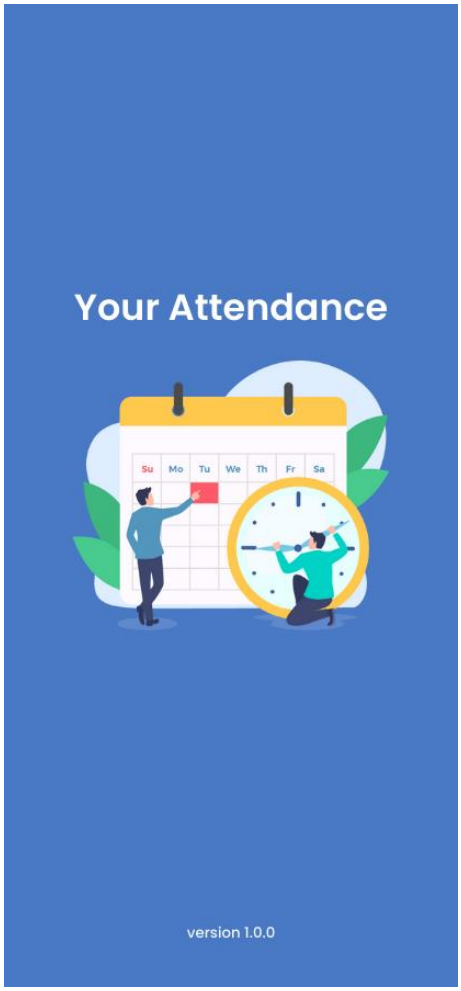
Nama Kolom	Tipe Data	Keterangan
id_address	integer	foreign key, not null
user_id	varchar(9)	primary key, foreign key, not null
attendance_date	datetime	primary key, not null
is_late	boolean	not null

Tabel 3.5 Tabel m_parameter

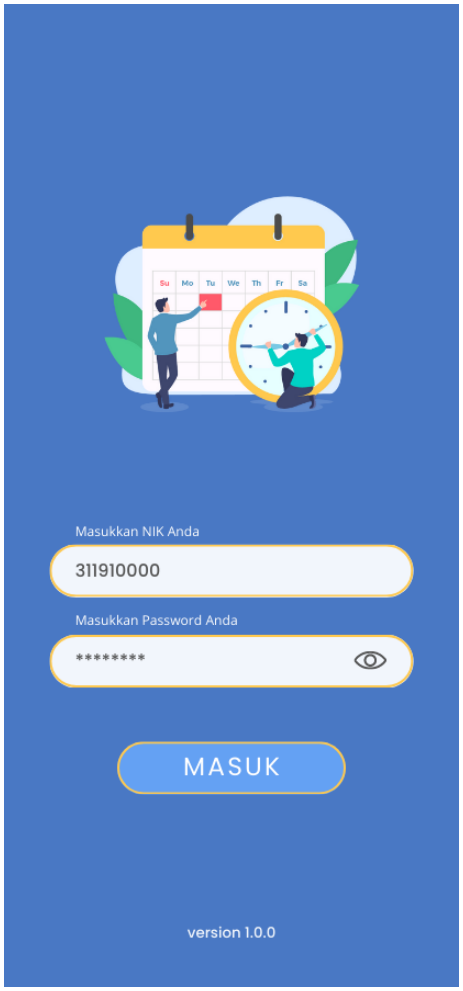
Nama Kolom	Tipe Data	Keterangan
parameter_id	varchar(20)	primary key, not null
seq	integer	not null
value	varchar(100)	not null
notes	varchar(100)	not null

3.4.4 Mock Up

Mock up merupakan rancangan *user interface* yang akan dibuat pada proyek. *Mock up* didesain untuk membantu perancangan setiap halaman dari aplikasi yang akan dibuat untuk menentukan setiap komponen, warna, serta informasi yang terdapat pada suatu halaman. Dengan adanya *mock up* dapat membantu menentukan desain yang sesuai dengan konsep aplikasi dan memberikan pengalaman menggunakan aplikasi yang nyaman bagi pengguna. *Mock up* yang dibuat ada *splash screen*, halaman login, halaman home, halaman kamera untuk mendaftarkan wajah dan absen, halaman riwayat absen, serta halaman untuk mengganti password. Gambar 3.8 sampai dengan gambar 3.18 menunjukkan *mock up* bagi setiap halaman pada aplikasi yang dikembangkan.

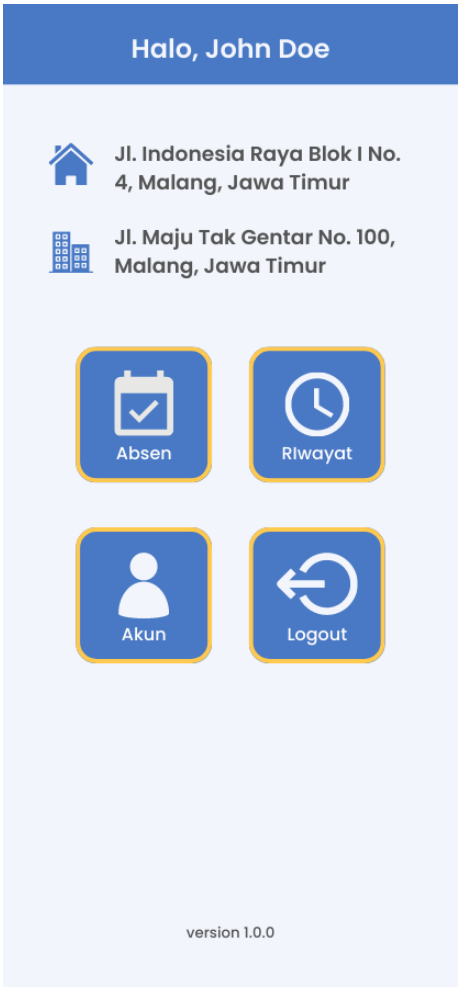


Gambar 3.8 *Splash Screen*



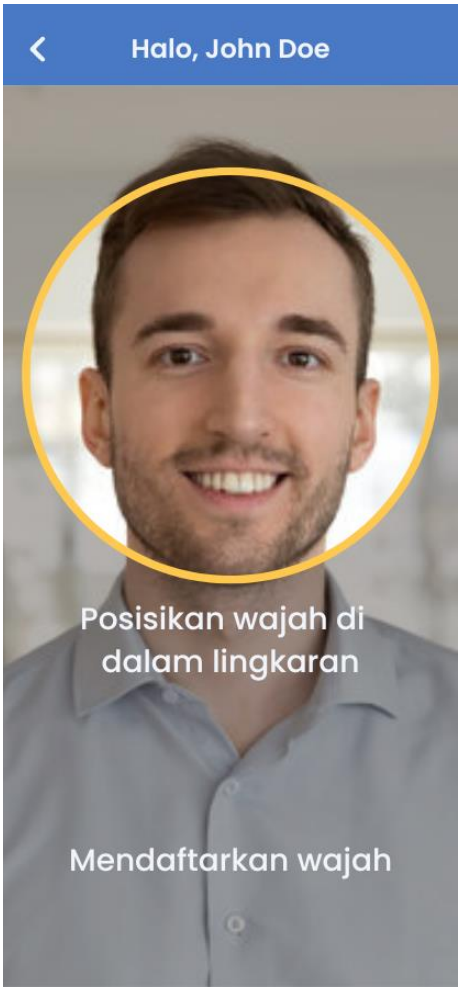
Gambar 3.9 Halaman Login

Gambar 3.8 menunjukkan *mock up* untul splash screen yang menjadi halaman untuk menyambut pengguna ketika pertama kali masuk ke aplikasi. Setelah splash screen akan dilanjutkan ke halaman login. Gambar 3.9 menunjukkan tampilan dari halaman login. Pada halaman login *user* dapat memasukkan ID dan password yang sudah terdaftar. Apabila sudah memasukkan ID dan password, *user* dapat menekan tombol masuk untuk mengakses fitur aplikasi lainnya. Jika ID juga password yang dimasukkan terdaftar pada *database* dan password yang dimasukkan benar, maka *user* akan diarahkan ke halaman beranda.

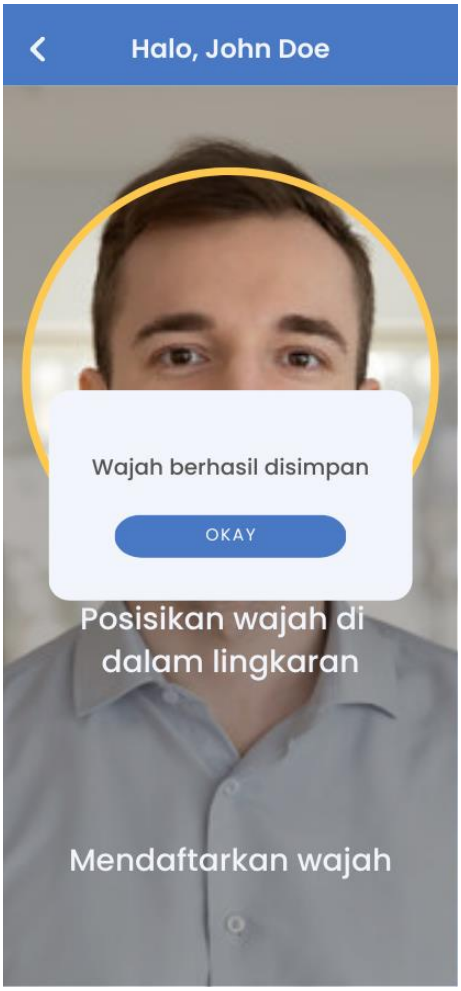


Gambar 3.10 Halaman Beranda

Apabila *user* berhasil masuk, akan diarahkan pada halaman beranda yang dapat dilihat pada gambar 3.10. Pada halaman ini terdapat toolbar yang menampilkan informasi nama *user* yang masuk. Selain itu terdapat informasi mengenai alamat kantor tempat pengguna bekerja dan alamat rumah milik pengguna. Kedua alamat inilah yang akan dicocokan pada lokasi pengguna ketika absen sesuai dengan jadwalnya. Kemudian terdapat empat tombol fitur yang dapat diakses *user*. Tombol absen akan mengarahkan *user* ke halaman absen pada gambar 3.13. Tombol riwayat akan mengarahkan *user* ke halaman riwayat pada gambar 3.17. Tombol akun akan mengarahkan *user* ke halaman ubah password pada gambar 3.18. Dan tombol *logout* akan mengeluarkan *user* dari halaman beranda, kembali pada halaman *login*.

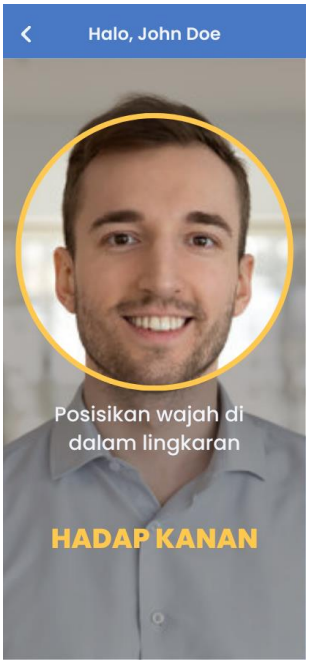


Gambar 3.11 Halaman Mendaftarkan Wajah

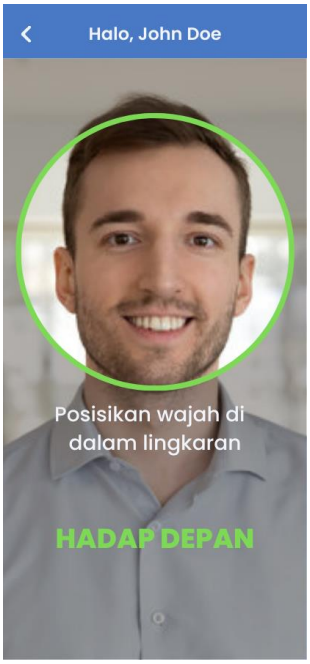


Gambar 3.12 Dialog Berhasil Mendaftarkan Wajah

Ketika pengguna pertama kali menggunakan aplikasi, maka ketika pengguna menekan tombol absen akan diarahkan pada halaman absen untuk mendaftarkan wajah seperti pada gambar 3.11. Gambar wajah pengguna akan diproses dan disimpan pada aplikasi. Dialog berhasil mendaftarkan wajah seperti gambar 3.12 akan muncul apabila sudah berhasil memproses dan menyimpan wajah *user*. Selanjutnya pengguna dapat kembali menekan tombol absen untuk melakukan proses presensi.



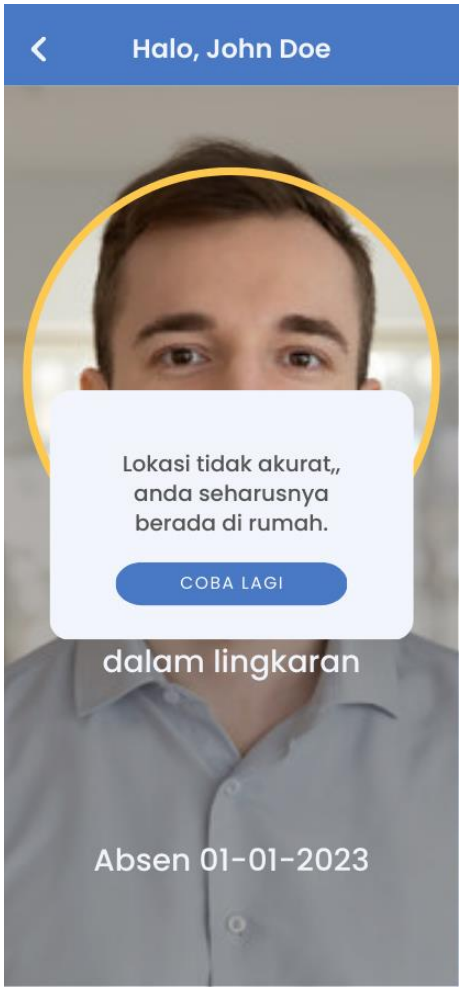
Gambar 3.13a Halaman Absen Hadap Kanan



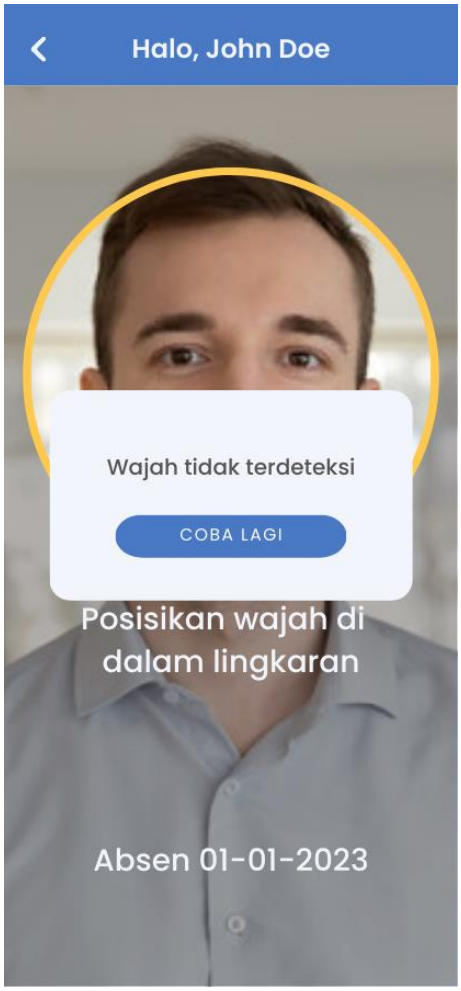
Gambar 3.13b Halaman Absen Hadap Depan

Gambar 3.14 Halaman Berhasil Absen

Apabila wajah *user* sudah terdaftar, maka ketika menekan tombol absen, *user* akan diarahkan pada halaman untuk absen seperti pada gambar 3.13a dan juga gambar 3.13b di mana terdapat instruksi yang perlu diikuti oleh pengguna. Pada halaman ini akan dilakukan proses *face detection*, *spoofing detection* dan *face recognition* untuk mengenali wajah *user*. Proses *face detection* akan mendeteksi wajah *user*. Jika sudah terdapat wajah yang terdeteksi, akan dilanjutkan pada proses *spoofing detection*. Proses *spoofing detection* akan memeriksa apakah arah wajah pengguna sesuai dengan perintah dari aplikasi. Apabila arah wajah sudah sesuai, akan lanjut pada proses *face recognition*. Proses *face recognition* akan membandingkan wajah *user* dengan yang sudah didaftarkan sebelumnya. Jika wajah yang terdaftar sesuai dengan wajah yang difoto, maka akan diarahkan ke halaman berhasil absen pada gambar 3.14. Pada halaman ini terdapat informasi apakah *user* absen tepat waktu atau terlambat, tanggal absen, waktu absen, serta lokasi absen.

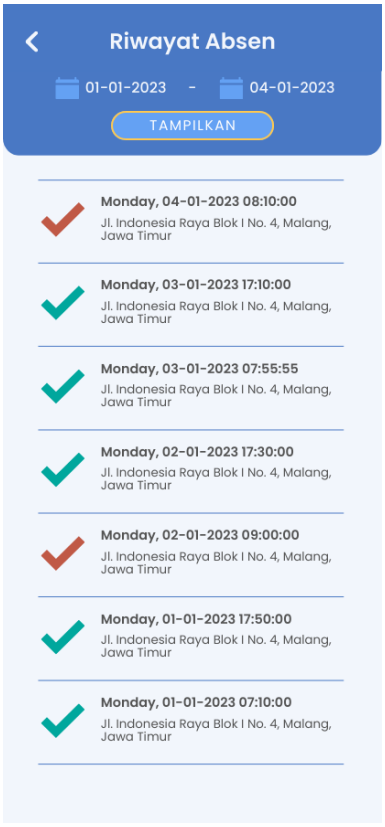


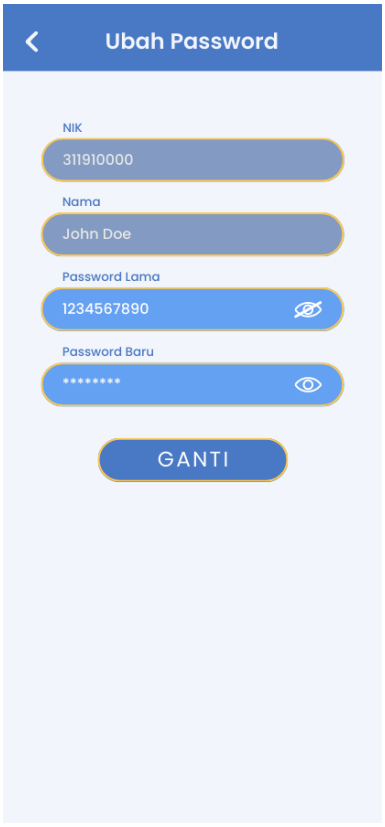
Gambar 3.15 Dialog Lokasi Salah



Gambar 3.16 Dialog Wajah Tidak Terdeteksi

Apabila wajah tersebut tidak sesuai dengan wajah yang terdaftar, maka akan tampil dialog menginformasikan bahwa wajahnya tidak dikenali. Gambar 3.16 menunjukkan dialog yang akan tampil bila wajah tidak dikenali. Selain itu juga ada proses pengecekan lokasi *user*. Jika *user* tidak berada pada jangkauan satu kilometer dari lokasi tempat *user* seharusnya berada, baik itu di kantor maupun di rumah, maka akan muncul dialog yang menginformasikan bahwa lokasi *user* tidak akurat. Gambar 3.15 menunjukkan dialog yang akan tampil apabila lokasi *user* tidak sesuai dengan lokasi yang seharusnya.





Gambar 3.17 Halaman Riwayat Absen Gambar 3.18 Halaman Ubah Password

Gambar 3.17 menunjukkan halaman riwayat absensi dari *user*. Pada halaman ini berisikan informasi riwayat absensi *user* dengan detail mengenai tanggal, jam, serta lokasi *user* melakukan absensi. *User* dapat memilih jarak waktu riwayat yang ingin ditampilkan dengan mengganti tanggal yang ada pada bagian atas dan menekan tombol untuk menampilan riwayat absensi sesuai dengan tanggal yang dipilih.

Gambar 3.18 menunjukkan halaman ubah password. Pada halaman ini memiliki empat *field*. *Field* ID dan nama berisikan informasi ID dan nama *user* yang otomatis terisi dan tidak dapat diubah. *Field password* lama dan *password* baru wajib diisi untuk mengubah *password user*. Pada field *password* lama dan *password* baru terdapat tombol yang dapat menyembunyikan atau menampilkan tulisan yang dimasukkan oleh *user*. Apabila *user* telah mengisi password lama dan *password* barunya, maka user dapat menekan tombol ganti untuk mengubah *password*.

3.5 Perancangan Pengujian

Pengujian aplikasi pada tugas akhir ini akan dibagi menjadi dua kategori. Yang pertama adalah kuesioner untuk pengujian kepuasan pengguna, pengujian ini akan melibatkan 5 partisipan menggunakan perangkat lunak masing-masing. Yang kedua adalah pengujian akurasi dari aplikasi, sama seperti pengujian sebelumnya, untuk pengujian kedua juga akan dilibatkan 5 partisipan. Setiap partisipan akan diminta untuk melakukan proses absen pada kondisi pencahayaan yang berbeda.

3.5.1 Rancangan Pengujian Kepuasan Pengguna

Pada pengujian kepuasa pengguna, 5 partisipan akan diminta untuk mencoba menggunakan seluruh fitur yang ada pada aplikasi dan memberikan pendapat mengenai pengalaman mengguna aplikasi tersebut. Fitur yang akan dicoba oleh pertisipan adalah login, mendaftarkan wajah, absensi, riwayat absensi, serta ganti password. Di mana pada fitur absensi sendiri partisipan akan mencoba *face detection*, *spoofing detection* dan juga *face recognition*.

Pertanyaan yang diberikan pada kuesioner merupakan pertanyaan seputar pengalaman dan kenyamanan pengguna ketika menggunakan aplikasi. Kuesioner terdiri atas 10 pernyataan, di mana partisipan diminta untuk memilih sangat tidak setuju, tidak setuju, biasa, setuju, dan sangat setuju dalam setiap pernyataan yang ada. Tabel 3.6 menunjukkan pertanyaan dan format jawaban untuk kuisioner yang akan diberikan pada setiap partisipan.

Tabel 3.6 Format Kuisioner Partisipan

No	Pernyataan	Sangat Tidak Setuju	Tidak Setuju	Biasa	Setuju	Sangat Setuju
1.	Tampilan halaman login menarik					
2.	Tampilan halaman home menarik					
3.	Tampilan halaman absen menarik					
4.	Tampilan halaman riwayat menarik					
5.	Tampilan halaman ganti password menarik					

6.	Proses login mudah dilakukan
7.	Proses pendaftaran wajah mudah dilakukan
8.	Proses absensi mudah dilakukan
9.	Proses ganti password mudah dilakukan
10.	Fitur riwayat absensi sangat membantu dalam melacak jejak absensi
11.	Fitur absensi sangat membantu dalam memudahkan pendataan absensi
12.	Fitur ganti password sangat membantu
13.	Aplikasi mudah untuk digunakan
14.	Mohon berikan saran dan/atau kritik

3.5.2 Rancangan Pengujian Akurasi

Pengujian yang berikutnya adalah pengujian akurasi. Pengujian akurasi dilakukan untuk melihat apakah aplikasi yang dikembangkan sudah akurat, efektif, dan siap untuk digunakan. Pengujian akurasi dilakukan pada proses absensi yang meliputi *face detection*, *face recognition*, dan *spoofing detection*. Yang diperhatikan adalah akurasi pengenalan wajah dan juga waktu yang dibutuhkan bagi partisipan untuk melakukan login hingga absensi tersimpan.

Pada pengujian akurasi, sepuluh partisipan akan diminta untuk melakukan proses absensi dengan berbagai kondisi. Perbedaan kondisi disesuaikan untuk melihat akurasi dari proses absensi, berikut adalah beberapa kondisi yang akan digunakan pada pengujian :

a. Intensitas cahaya tinggi dan kamera resolusi tinggi.
b. Intensitas cahaya sedang dan kamera resolusi tinggi.
c. Intensitas cahaya rendah dan kamera resolusi tinggi.
d. Intensitas cahaya tinggi dan kamera resolusi rendah.

e. Intensitas cahaya sedang dan kamera resolusi rendah.

f. Intensitas cahaya rendah dan kamera resolusi rendah.

Tolak ukur tinggi, sedang, dan rendah memiliki banyak definisi dan dapat mempengaruhi hasil pengujian akurasi. Oleh karena itu akan diberi tolak ukur yang menentukan ukuran tinggi, sedang, dan rendah yang digunakan pada pengujian proyek tugas akhir ini. Perhitungan tolak ukur intensitas cahaya menggunakan aplikasi *Light Meter*.

a. Intensitas cahaya tinggi

   Kondisi pada siang hari atau ruangan pencahayaan tinggi dengan nilai fluks diatas 150 lux.

b. Intensitas cahaya sedang

   Kondisi pada pagi hari ketika matahari terbit, sore hari ketika matahari terbenam, atau ruangan dengan pencahayaan cukup dengan nilai fluks diantara 40 lux sampai 100 lux.

c. Intensitas cahaya rendah

   Kondisi pada malam hari atau ruangan dengan sedikit pencahayaan dengan nilai fluks dibawah 10 lux.

d. Kamera resolusi tinggi

   Kondisi di mana perangkat yang digunakan memiliki kamera depan dengan resolusi 16 megapixel. Perangkat yang akan digunakan pada penelitian ini adalah Vivo T1 5G.

e. Kamera resolusi rendah

   Kondisi di mana perangkat yang digunakan memiliki kamera depan dengan resolusi 5 megapixel. Perangkat yang akan digunakan pada penelitian ini adalah Oppo A12.

Tabel 3.7 menunjukkan tabel yang akan diisi ketika pelaksanaan pengujian proses absensi. Kolom kondisi akan mewakili setiap kondisi yang akan diuji, kolom partisipan akan berisikan nama dari setiap partisipan, kolom berhasil atau gagal untuk mencatat keberhasilan absensi, dan kolom keterangan untuk memberi keterangan kegagalan absensi.

Tabel 3.7 Rancangan Pengujian Absensi

Partisipan	Kondisi	Berhasil / gagal	Keterangan

Tahap berikutnya setelah ditemukan kondisi yang paling sesuai untuk melakukan absensi adalah melakukan pengujian menggunakan wajah orang lain, di mana partisipan lain akan mencoba untuk melakukan absensi dengan aplikasi yang sudah memiliki wajah partisipan lainnya yang terdaftar. Juga akan dilakukan pengujian menggunakan *video call*, di mana dalam proses absensi, partisipan akan mencoba untuk melakukan absensi dengan melalui *video call*.

Pengujian akurasi yang terakhir dilakukan untuk menguji kemampuan mendeteksi lokasi pengguna. Yaitu dengan mencoba melakukan proses presensi dengan mengaktifkan aplikasi untuk memberikan lokasi palsu. Pada pengujian ini akan menggunakan aplikasi Fake GPS.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Sistem Presensi dan Model Pengenalan Wajah

Sistem presensi pada aplikasi ini akan melalui tiga proses, yaitu mendeteksi wajah, *spoofing detection*, dan juga pengenalan wajah. Proses mendeteksi wajah akan mencari wajah dari hasil gambar kamera secara *real time* kemudian memotong gambar wajah pengguna. Proses ini dilakukan untuk mencari wajah yang akan diproses juga sebagai data *input* pada proses pengenalan wajah. Proses mendeteksi wajah menggunakan *Machine Learning Kit* (ML Kit). ML Kit sendiri menggunakan metode BlazeFace. Setelah berhasil mendeteksi wajah, maka proses akan dilanjutkan dengan *spoofing detection*.

S*poofing detection* merupakan proses di mana aplikasi akan meminta pengguna untuk menghadap ke atas, bawah, kiri, atau kanan secara acak. Proses ini dilakukan untuk menghindari adanya kecurangan menggunakan foto ketika melakukan presensi. Apabila arah wajah pengguna benar dan wajah tersebut sesuai dengan pemilik akun tersebut, maka proses presensi berhasil. Tetapi meskipun wajah pengguna sesuai dengan pemilik akun, namun arah wajah pengguna salah maka proses presensi akan gagal. Sama seperti proses mendeteksi wajah, *spoofing detection* yang mendeteksi arah wajah pengguna juga menggunakan ML Kit yang menggunakan model BlazeFace.

Proses terakhir adalah pengenalan wajah. Hasil potongan gambar wajah pengguna pada proses pendeteksi wajah akan diproses menggunakan model FaceNet. Model ini dibangun dengan menggunakan tensorflow, kemudian dikonversi menjadi Tensorflow Lite. Model ini dikonversi menjadi Tensorflow Lite agar ukurannya menjadi lebih kecil dan lebih efisien untuk dijalankan pada aplikasi *mobile*. Model ini merupakan model *pretrained* yang hanya membutuhkan satu *sample* untuk dibandingkan. Perbandingkan wajah pengguna dengan *sample* menggunakan metode *cosine similarity*.

4.2 Perancangan Aplikasi

Aplikasi presensi ini memiliki tiga fitur utama, yaitu presensi, ganti password, dan riwayat presensi. Ketiga fitur ini dibangun untuk mencapai tujuan daru Tugas Akhir ini, yaitu membuat suatu aplikasi presensi yang dapat mengenali wajah dan membantu pendataan presensi ketika *hybrid working*.

4.2.1 Lokasi Pengguna

Untuk mencapai tujuan dari Tugas Akhir ini, aplikasi dibuat agar dapat membandingkan lokasi pengguna dengan lokasi di mana pengguna seharusnya berada sesuai dengan jadwal kerja pengguna pada proses presensi. Hal ini untuk memastikan bahwa selama pengguna *Work From Home* (WFH) atau *Work From Office* (WFO), pengguna berada pada lokasi yang tepat, pengguna berada di kantor ketika WFO dan pengguna berada di rumah ketika WFH. Dalam mencegah pengguna untuk menyesuaikan alamat yang ada dengan lokasi pengguna saat ini, alamat rumah dan kantor pengguna akan disimpan pada *database webservice* dan tidak dapat diubah sendiri oleh pengguna.

Perbandingan lokasi pengguna saat ini dengan lokasi rumah atau kantor pengguna diberikan toleransi sebanyak 1000 meter. Apabila lokasi pengguna saat ini melebihi jarak 1000 meter dibandingkan dengan lokasi rumah atau kantor pengguna sesuai jadwalnya, maka pengguna tidak dapat melakukan presensi. Namun apabila kurang dari itu maka proses presensi akan dilanjutkan pada proses pendeteksi wajah.

Perhitungan jarak antara lokasi pengguna dengan lokasi rumah atau kantor pengguna dilakukan dengan mengitung jarak koordinat kedua lokasi. Pertama aplikasi akan mengambil garis lintang dan garis bujur lokasi saat ini setelah mendapatkan ijin pengguna untuk mengakses lokasi. Kemudian koordinat tersebut akan dikirimkan pada *webservice*. *Webservice* akan memeriksa jadwal kerja pengguna dan mengambil koordinat rumah atau kantor sesuai dengan jadwal kerja tersebut.

```
001  public double getDistanceInKM(double lat1, double long1,
     double lat2, double long2) {
002       double _eQuatorialEarthRadius = 6378.1370D;
003       double _d2r = (Math.PI / 180D);
004       double dlong = (long2 - long1) * _d2r;
005       double dlat = (lat2 - lat1) * _d2r;
006       double a = Math.pow(Math.sin(dlat / 2D), 2D) +
          Math.cos(lat1 * _d2r) * Math.cos(lat2 * _d2r) *
          Math.pow(Math.sin(dlong / 2D), 2D);
007       double c = 2D * Math.atan2(Math.sqrt(a), Math.sqrt(1D -
          a));
008       double d = _eQuatorialEarthRadius * c;
009       return d;
010  }
```

Gambar 4.1 Potongan Kode Hitung Jarak

Gambar 4.1 menunjukkan fungsi untuk menghitung jarak akan menghasilkan selisih antara koordinat lokasi pengguna dengan lokasi yang terdaftar. Jarak tersebut kemudian akan dibandingkan dengan maksimal jarak yang sudah ditentukan sebelumnya. Baris 002 merupakan variabel yang berisikan besar radius dari bumi. Baris 003 merupakan variabel yang menampung hasil π dibagi dengan 180. Baris 004 dan 005 menghitung perbedaan antara kedua koordinat longitude dan latitude. Baris 006 sampai 008 mengaplikasikan rumus untuk menghitung jarak antara kedua koordinat longitude dan latitude tersebut. Pada perancangan aplikasi ini, toleransi jarak yang diberikan sebesar 1000 meter. Apabila jaraknya kurang dari maksimal jarak, maka halaman presensi akan menampilkan kamera untuk masuk ke proses selanjutnya, yaitu proses mendeteksi wajah.

4.2.2 Deteksi Wajah, *Spoofing Detection*, dan Pengenalan Wajah

Pada proses mendeteksi wajah, halaman akan menampilkan kamera yang disertai dengan *bounding box* sebagai petunjuk bagi pengguna untuk memosisikan wajahnya. *Bounding box* bertujuan agar wajah pengguna tidak terlalu jauh maupun terlalu dekat. Bila aplikasi belum dapat mendeteksi wajah, terdapat tulisan “Face Not Found”. Jika wajah sudah terdeteksi maka tulisan akan berubah dan proses dilanjutkan pada *spoofing detection*.

Spoofing detection memiliki tujuan untuk mencegah kecurangan oleh pengguna ketika melakukan presensi dengan menggunakan foto wajah pengguna. Proses *spoofing detection* dimulai dengan aplikasi memberikan perintah secara acak

untuk menghadap ke atas, bawah, kiri, atau kanan. Perintah diberikan berupa tulisan yang sebelumnya “Face Not Found” menjadi “HADAP KANAN” sesuai dengan arah yang terpilih. Setelah pengguna menghadap pada suatu arah, maka tulisan akan berubah menjadi “HADAP DEPAN” untuk meminta pengguna kembali menghadap ke kamera. Apabila pengguna sudah menghadap depan maka aplikasi akan mengambil foto wajah pengguna untuk dilanjutkan pada proses pengenalan wajah.

Proses pengenalan wajah diawali dengan memotong gambar wajah pengguna. Kemudian potongan wajah tersebut akan diproses dengan model FaceNet. Hasil proses dengan model FaceNet berupa *signature* atau fitur dari wajah pengguna. Aplikasi akan memeriksa apakah terdapat file *sample signature* pengguna yang tersimpan pada aplikasi. Apabila pengguna belum pernah mendaftarkan wajahnya pada aplikasi, maka *signature* tersebut akan disimpan sebagai file JSON. Namun apabila pengguna sudah pernah mendaftarkan wajahnya, maka file JSON akan dibaca sebagai *sample* dan dibandingkan dengan *signature* foto wajah pengguna saat ini. Signature dibandingkan dengan menghitung *cosine similarity* kedua wajah.

```
001   double cosineSimilarity(List<List<double>> embeddings) {
002       List<double> embeddings1 = embeddings[0];
003       List<double> embeddings2 = embeddings[1];
004       double dot = 0.0;
005       double nA = 0.0;
006       double nB = 0.0;
007       for (int i = 0; i < embeddings1.length; i++) {
008           dot += (embeddings1[i] * embeddings2[i]);
009           nA += pow(embeddings1[i], 2.0);
010           nB += pow(embeddings2[i], 2.0);
011       }
012       double distance = 1 - (dot / (sqrt(nA) * sqrt(nB)));
013       return min(1, max(0, distance));
014   }
```

Gambar 4.2 Potongan Kode Menghitung *Cosine Similarity*

Gambar 4.2 menunjukkan fungsi untuk menghitung *cosine similarity*. Pertama, pada fungsi tersebut akan ada satu parameter yang berisikan data yang ingin dibandingkan. Kedua, pada baris 002 dan 003 data tersebut akan dimasukkan kedalam dua variabel berbeda, yaitu “embeddings1” dan “embeddings2”. Selanjutkan dibuat variabel yang dibutuhkan untuk melakukan perhitungan *cosine similarity*. Variabel “dot” berisikan penjumlahan dari hasil perkalian

“embeddings1” dan “embeddings2”. Variabel “nA” berisikan penjumlahan hasil pangkat dua “embeddings1”. Variabel “nB” berisikan penjumlahan hasil pangkat dua “embeddings2”. Kemudian dihitung jarak atau kemiripan dari keduanya.

4.2.3 Ganti Password

Fitur ganti password dibuat agar pengguna dapat mengubah password *default* menjadi password yang diinginkan. Hal ini karena pengguna tidak dapat mendaftarkan akun sendiri, melainkan akan dibuatkan dengan password *default*. Oleh karena itu dibutuhkan fitur ganti password di mana pengguna dapat mengubah passwordnya menjadi unik sesuai dengan keinginan pengguna untuk mencegah pengguna lain masuk ke akun pengguna.

Pada halaman ganti password terdapat informasi NIK dan juga nama pengguna yang tidak dapat diubah. Juga *field* bagi pengguna untuk mengisi password lama dan password baru yang diinginkan. Apabila password lama tidak sesuai, maka proses ganti password akan gagal. Namun bila password lama sudah sesuai, proses ganti password berhasil. Akan muncul *toast* dibagian bawah halaman yang menginformasikan apakah ganti password tersebut gagal atau berhasil.

4.2.4 Riwayat Presensi

Fitur riwayat presensi merupakan fitur di mana pengguna dapat melihat hasil dari riwayat presensi yang sudah dilakukan. Pada halaman riwayat presensi, terdapat tanggal yang dapat dipilih oleh pengguna. Maksimal jangka tanggal awal dan akhir yang dapat dipilih pengguna adalah satu minggu atau tujuh hari. Secara default tanggal akan menampilkan satu minggu terakhir dari tanggal hari ini. Kemudian dibawahnya terdapat tombol “TAMPILKAN” untuk menampilkan data presensi sesuai dengan tanggal yang terpilih.

Data tanggal yang ditampilkan terdapat hari, tanggal, waktu, dan juga alamat di mana pengguna tersebut melakukan presensi. Pada bagian kanan terdapat gambar yang akan berwarna merah ketika pengguna terlambat ketika melakukan presensi tersebut dan berwarna hijau ketika pengguna tepat waktu.

4.3 Hasil Pengujian

Pengujian pada Tugas Akhir ini akan dilakukan dalam tiga tahap. Tahap pertama adalah pengujian fungsionalitas. Pengujian fungsionalitas dilakukan untuk mengetahui bagaimana aplikasi berjalan pada beberapa skenario pengujian aplikasi. Kemudian dilanjukan dengan pengujian pengenalan wajah untuk mengetahui performa model pengenalan wajah pada dalam beberapa kondisi ruangan dan perangkat. Dan yang terakhir adalah pengujian kepuasan pengguna. Pengujian kepuasan pengguna dilakukan untuk mengetahui kepuasan pengguna terhadap fitur dan tampilan aplikasi tersebut. Pada pengujian ini akan ada lima partisipan yang ikut serta untuk menguji aplikasi ini.

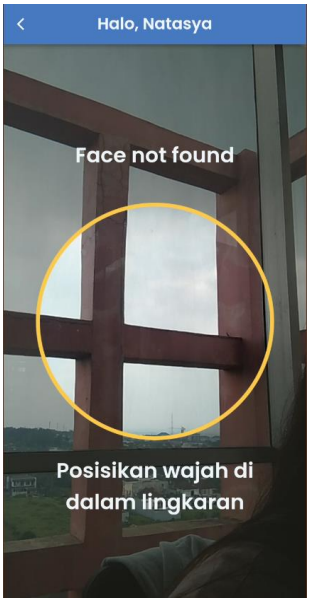
4.3.1 Hasil Pengujian Fungsionalitas Fitur Presensi

Pengujian fungsionalitas akan menguji seluruh fitur dari aplikasi yang sudah dibuat. Setiap fitur akan dicoba dan ditunjukkan untuk lebih memahami alur presensi, ganti password, dan juga riwayat presensi pada aplikasi dengan beberapa skenario atau kondisi. Selain lebih memahami alur, juag mencoba beberapa skenario yang kemungkinan akan terjadi. Fitur pertama yang akan diuji fungsionalitasnya adalah fitur presensi.

Ketika pertama kali masuk halaman presensi, aplikasi akan menampilkan halaman *loading* secara otomatis. Kemudian aplikasi akan mengambil garis lintang dan bujur perangkat dan menyocokkannya dengan garis lintang dan bujur yang tersimpan pada *database*. Apabila alamat sudah sesuai, maka akan masuk ke halaman presensi seperti pada gambar 4.4. Tetapi apabila alamat belum sesuai, maka akan menampilkan dialog yang menginformasikan pengguna berada di lokasi yang salah seperti pada gambar 4.3. Pada dialog tersebut terdapat tombol “COBA LAGI”, pengguna dapat menekan tombol “COBA LAGI” untuk kembali memproses alamat pengguna.

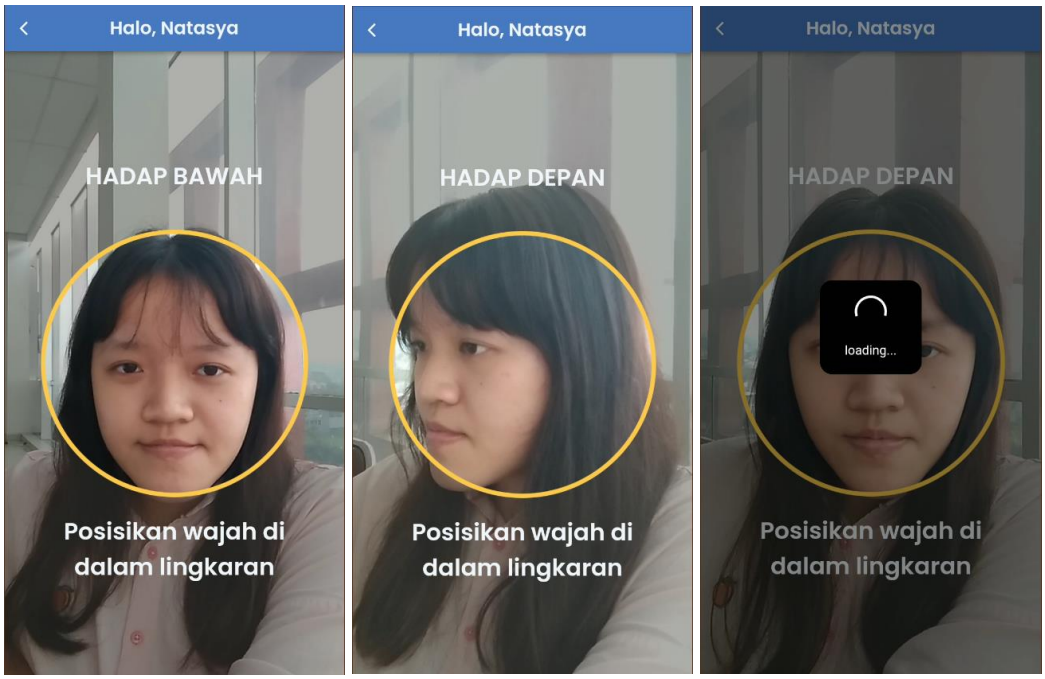
Gambar 4.3 Halaman Presensi Lokasi Tidak Sesuai

Gambar 4.4 menunjukkan halaman presensi setelah pemeriksaan alamat. Tulisan “Face Not Found” menunjukkan bahwa aplikasi belum bisa menemukan wajah untuk diidentifikasi. Tulisan ini akan berubah ketika aplikasi sudah berhasil menemukan wajah. Selain itu, pada bagian bawah juga terdapat tulisan untuk menginstruksikan pengguna agar memosisikan wajahnya pada lingkaran yang ada.



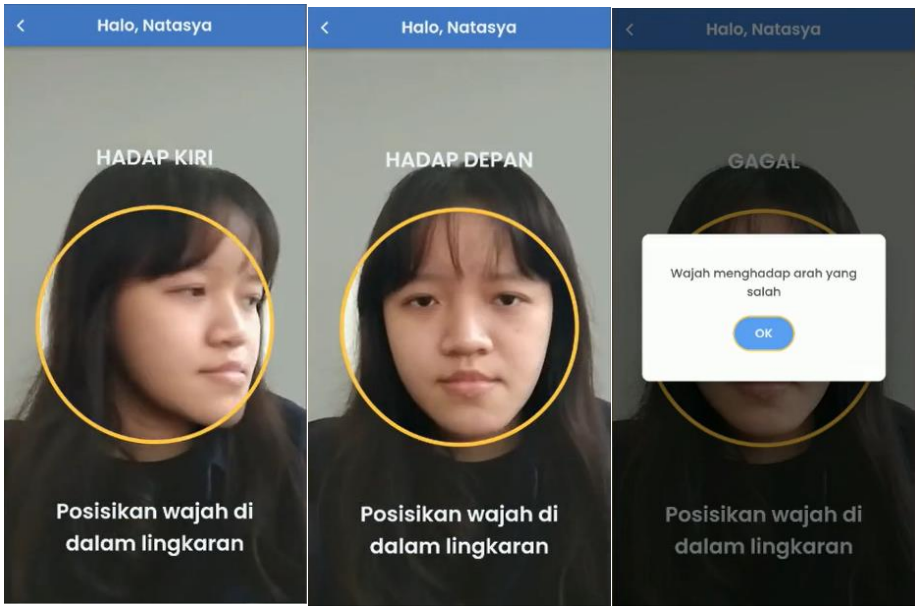
Gambar 4.4 Halaman Presensi

Pada gambar 4.5 menunjukkan halaman presensi yang menampilkan perintah pada pengguna untuk menghadap suatu arah secara random. Apabila pengguna telah menghadap ke suatu arah, terlepas dari sesuai atau tidaknya arah wajah pengguna, maka tulisan tersebut berubah menjadi "HADAP DEPAN" agar pengguna kembali menghadap ke arah kamera. Setelah menghadap ke depan, aplikasi akan memproses wajah pengguna dan menampilkan halaman *loading*. Jika arah wajah pengguna sudah sesuai dengan perintah, maka aplikasi akan melakukan *face recognition*.



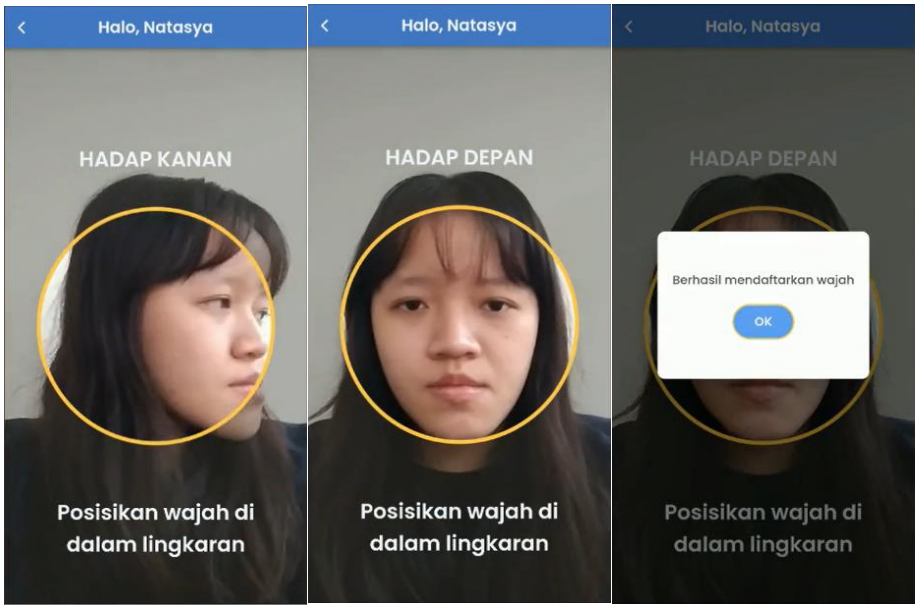
Gambar 4.5 Proses *Spoofing Detection*

Gambar 4.6 menunjukkan skenario di mana arah wajah pengguna tidak sesuai dengan instruksi yang diberikan oleh aplikasi. Aplikasi menginstruksikan pengguna untuk menghadap ke arah kiri sedangkan pengguna menghadap ke arah kanan. Karena arah wajah yang tidak sesuai, akan muncul dialog yang memberitahu pengguna bahwa arah wajah yang dilakukan tidak sesuai. Oleh karena itu proses *spoofing detection* gagal dan data presensi tidak akan disimpan. Pengguna dapat mencoba lagi dengan mengulangi proses yang sama.



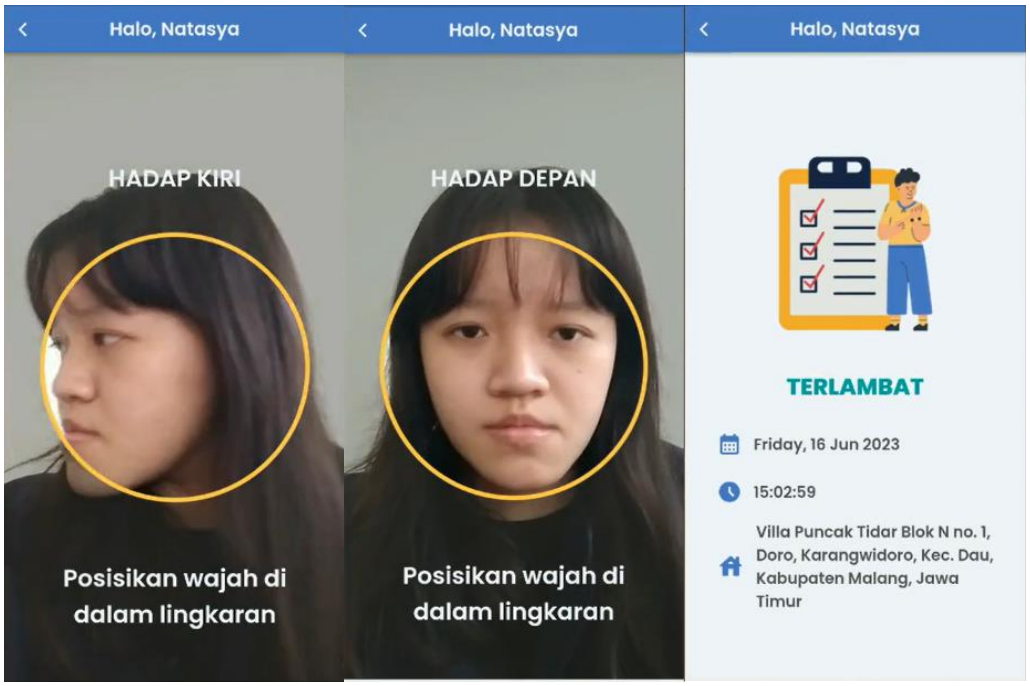
Gambar 4.6 Proses Presensi Arah Tidak Sesuai

Gambar 4.7 menunjukkan skenario di mana arah wajah pengguna sesuai dengan instruksi. Karena pengguna baru pertama kali melakukan proses ini, maka wajah pengguna akan disimpan dalam aplikasi sebagai sample untuk dibandingkan dengan wajah ketika proses presensi selanjutnya. Untuk menandakan keberhasilan menyimpan wajah pengguna, aplikasi akan menampilkan dialog sukses.



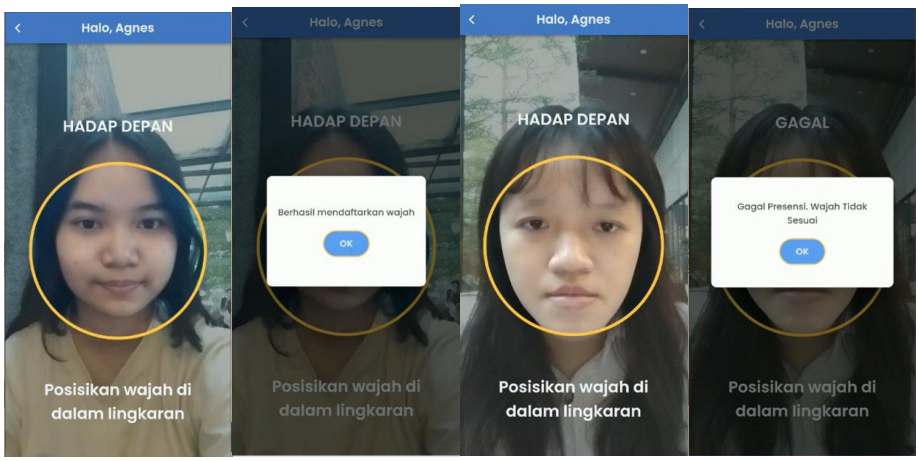
Gambar 4.7 Proses Presensi Daftar Wajah

Gambar 4.8 menunjukkan skenario di mana pengguna yang sudah mendaftarkan wajahnya sebelumnya melakukan presensi lagi. Arah wajah pengguna yang sesuai dan juga wajah yang sesuai dengan wajah yang didaftarkan berarti proses presensi berhasil dan data presensi pengguna akan dikirimkan ke *webservice* untuk disimpan. Proses presensi yang berhasil akan mengarahkan pengguna ke halaman sukses. Pada halaman sukses ini terdapat informasi keterlambatan pengguna. Pengguna akan dianggap terlambat apabila waktu absen pengguna lebih dari waktu masuk atau kurang dari waktu pulang yang disimpan pada *database*.



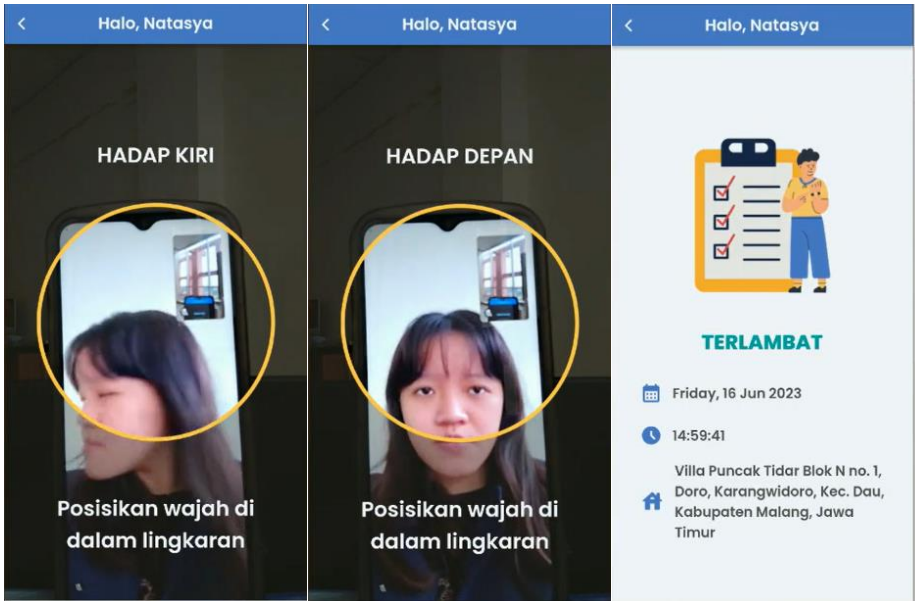
Gambar 4.8 Proses Presensi Berhasil

Gambar 4.9 menunjukkan proses presensi di mana wajah pengguna tidak sesuai dengan wajah yang terdaftar pada aplikasi. Pada aplikasi ini didaftarkan wajah pengguna “Agnes”, kemudian pengguna “Natasya” mencoba untuk melakukan presensi. Ketika pengguna “Natasya” melakukan presensi, akan muncul dialog yang menginfokan pengguna bahwa wajah tidak sesuai dan data presensi tidak akan disimpan.



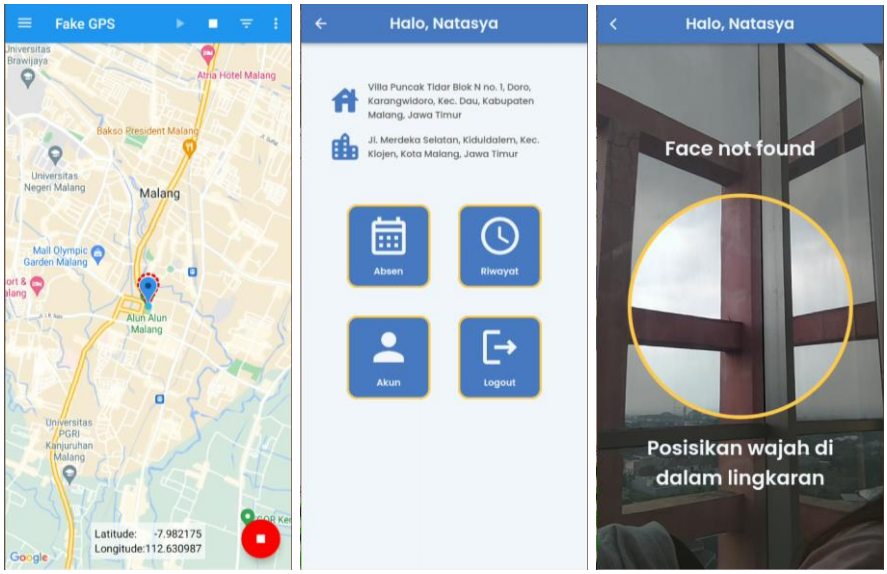
Gambar 4.9 Proses Presensi Wajah Tidak Sesuai

Pada pengujian fungsionalitas ini juga akan dilakukan pengujian dengan skenario tindak kecurangan dengan menggunakan *video call*. Pada skenario ini, pengguna akan mencoba untuk melakukan presensi dengan melakukan *video call*, kamera perangkat presensi akan dihadapkan dengan perangkat lain yang menampilkan wajah pengguna ketika *video call*. Dilakukan pengujian dengan menggunakan video call agar pengguna tetap dapat melakukan instruksi yang diminta aplikasi secara acak. Dapat dilihat bahwa ketika melakukan presensi dengan *video call*, presensi berhasil. Gambar 4.10 menunjukkan hasil ketika presensi menggunakan *video call*.



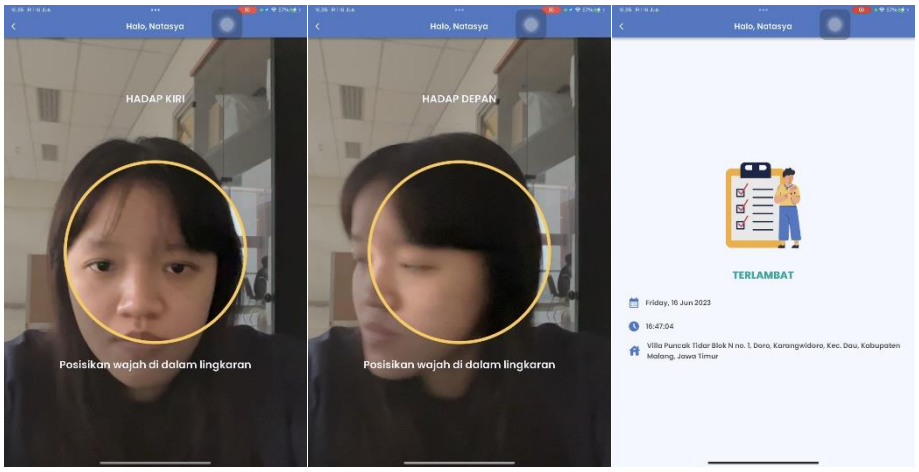
Gambar 4.10 Proses Presensi dengan *Video Call*

Pengujian fungsionalitas terhadap fitur presensi juga akan dicoba dengan mencoba mengunduh aplikasi *Fake GPS*. Pengujian ini dilakukan untuk melihat apakah aplikasi ini dapat mengambil koordinat lokasi pengguna dengan baik meskipun terdapat aplikasi lain yang memberikan lokasi palsu. Pada pengujian ini, lokasi yang terdaftar bagi pengguna adalah Alun Alun Malang, dan lokasi pengguna sebenarnya adalah Universitas Ma Chung. Aplikasi *Fake GPS* akan diatur agar lokasi pengguna saat ini sesuai dengan lokasi yang terdaftar pada *database*, yaitu berada di Alun Alun Malang. Jarak antara Alun Alun Malang (-7.982412619346088, 112.63085421182653) dengan Universitas Ma Chung (-7.956846560003925, 112.58978696764544) adalah kurang lebih enam kilometer, sedangkan jarak toleransi maksimal aplikasi adalah satu kilometer, sehingga seharusnya proses presensi tidak dapat dilakukan karena lokasi yang berbeda antara lokasi sebenarnya pengguna saat ini (Universitas Ma Chung) dengan lokasi yang terdaftar pada *database* (Alun Alun Malang). Dapat dilihat pada gambar 4.9 bahwa ketika melakukan presensi dengan aplikasi *Fake GPS*, proses presensi tetap dapat dilakukan. Oleh karena itu, meskipun lokasi pengguna tidak sesuai dengan yang terdaftar pada database, tetapi karena menggunakan aplikasi *Fake GPS*, presensi tetap dapat dilakukan dan tidak terdeteksi bahwa lokasi pengguna salah. Gambar 4.11 menunjukkan proses dan hasil ketika melakukan presensi dengan *Fake GPS*.



Gambar 4.11 Proses Presensi dengan *Fake GPS*

Pengembangan aplikasi berbasis Flutter membuat aplikasi ini dapat dijalankan pada perangkat dengan sistem operasi Android maupun IOS. Oleh karena itu, dilakukan pengujian berjalannya aplikasi pada perangkat dengan basis IOS. Pengujian ini akan dilakukan dengan menggunakan perangkat IPad Mini 6. Gambar 4.12 menunjukkan hasil ketika aplikasi dijalankan pada perangkat tersebut. Dapat dilihat bahwa aplikasi tetap dapat menjalankan fitur presensi dengan baik pada perangkat dengan basis IOS.



Gambar 4.12 Proses Presensi dengan Perangkat IOS

Terdapat kendala ketika menjalankan aplikasi pada perangkat IOS karena didapati perbedaan format antara perangkat IOS dengan perangkat Android. Di mana pada perangkat IOS format gambarnya adalah BGRA8888, sedangkan pada perangkat Android format gambarnya adalah YUV420. Oleh karena itu ditambahkan kondisi untuk membedakan kode yang digunakan ketika menggunakan perangkat IOS dan perangkat Android. Gambar 4.13 Menunjukkan potongan kode yang ditambahkan untuk membedakan hal tersebut. Pertama, pada baris 001 kode tersebut akan memeriksa apakah platform yang digunakan memiliki sistem operasi Android. Jika benar maka baris 002 akan memanggil fungsi untuk memproses format gambar YUV420. Namun jika tidak, pada baris 003 akan diperiksa apakah platform yang digunakan adalah IOS. Jika benar maka pada baris 004 akan memanggil fungsi untuk memproses format gambar BGRA8888.

```
001  if (Platform.isAndroid) {

002       fileImageBefore = convertCameraImage(cameraImage);
003  } else if (Platform.isIOS) {
004       fileImageBefore = await convertBGRA8888(cameraImage);
005  }
```

Gambar 4.13 Potongan Kode Tambahan untuk Format Gambar Berbeda

Kendala lainnya adalah ketika perangkat IOS mengambil gambar menggunakan kamera depan, hasil gambarnya akan secara otomatis dibalik secara vertikal. Hal ini menyebabkan arah wajah pengguna yang sebelumnya kanan menjadi menghadap kiri, demikian pula sebaliknya. Oleh karena itu diberikan tambahan kode untuk mengubah arah wajah yang dideteksi, sehingga ketika perintah memberikan arahan untuk menghadap kanan, maka aplikasi akan memeriksa hasil gambar dengan syarat yang menghadap kiri. Gambar 4.14 menunjukkan potongan kode untuk mengatasi hal ini. Pertama pada baris 001 akan diperiksa apakah pengguna menggunakan platform IOS, jika iya maka arah wajah akan diubah dari kiri menjadi kanan atau kanan menjadi kiri.

```
001   if (Platform.isIOS) {
002       if (_facing.compareTo("KIRI") == 0) {
003            _facing = "KANAN";
004       } else if (_facing.compareTo("KANAN") == 0) {
005            _facing = "KIRI";
006       }
007   }
```

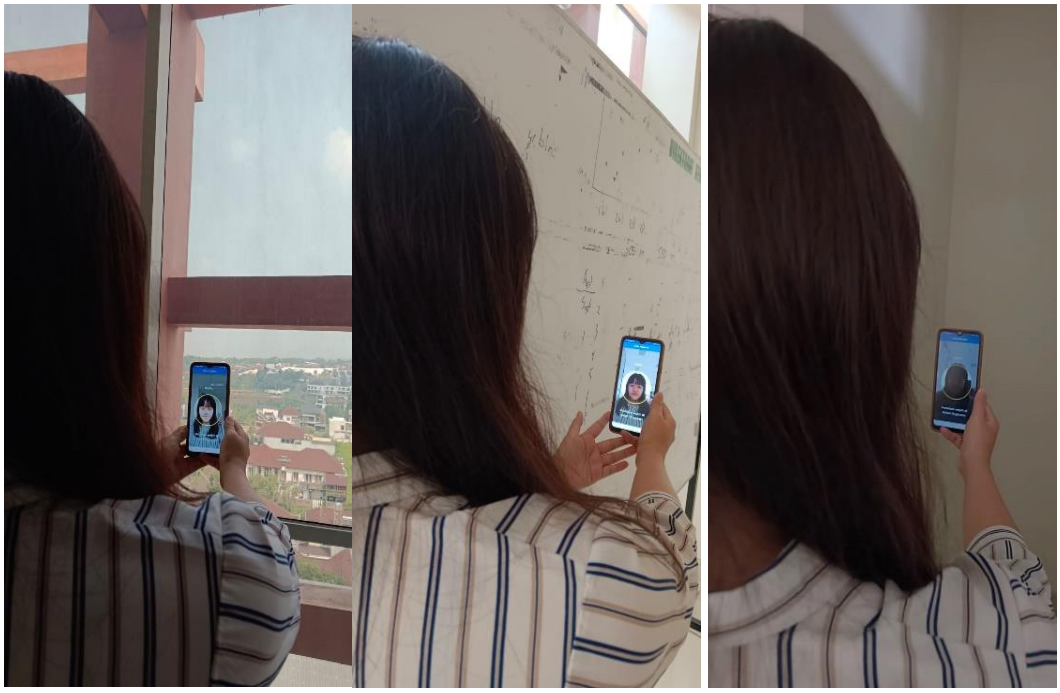
Gambar 4.14 Potongan Kode Tambahan untuk Hasil Gambar IOS Terbalik

4.3.2 Hasil Pengujian Pengenalan Wajah

Pengujian pengenalan wajah dilakukan untuk memeriksa performa model pengenalan wajah pada beberapa kondisi pencahayaan dan resolusi kamera yang berbeda beda. Pengujian akan dilakukan oleh sepuluh partisipan. Setiap partisipan akan mencoba melakukan presensi dengan enam kondisi yang telah dijelaskan sebelumnya pada partisipan.

Gambar 4.15 bagian kiri menunjukkan kondisi pengujian dengan pencahayaan tinggi, bagian tengah menunjukkan kondisi pengujian dengan pencahayaan sedang, dan bagian kanan menunjukkan kondisi pengujian dengan

pencahayaan rendah. Pada setiap kondisi pencahayaan akan dilakukan pengujian dengan dua perangan yang memiliki resolusi kamera berbeda. Perangkat dengan resolusi kamera tinggi akan menggunakan Vivo T1 5G yang memiliki kamera resolusi 12 megapixel. Perangkat dengan resolusi kamera rendah akan menggunakan Oppo A12 yang memiliki kamera resolusi 5 megapixel.



Gambar 4.15 Kondisi Pencahayaan Pengujian

Tabel 4.15 menunjukkan hasil pengujian model pengenalan wajah dengan kondisi pencahayaan terang (di atas 150 lux). Pada pengujian ini setiap partisipan melakukan pengujian di lokasi yang sama dengan perangkat yang sama. Dapat dilihat bahwa dalam kondisi pencahayaan terang dan kamera resolusi tinggi maupun rendah seluruh partisipan dapat dikenali. Oleh karena itu dengan kondisi cahaya tinggi, kedua resolusi kamera tinggi dan rendah dapat mengidentifikasi wajah partisipan dengan baik dengan tingkat keberhasilan 100% juga dapat mendeteksi wajah dan melakukan *spoofing detection* dengan tingkat akurasi 100%.

Tabel 4.1 Hasil Pengujian Pengenalan Wajah Pencahayaan Terang

No	Subjek	*Spoofing Detection*		Pengenalan Wajah	
		Kamera Resolusi Tinggi	Kamera Resolusi Rendah	Kamera Resolusi Tinggi	Kamera Resolusi Rendah
1	Subjek 1	V	V	V	V
2	Subjek 2	V	V	V	V
3	Subjek 3	V	V	V	V
4	Subjek 4	V	V	V	V
5	Subjek 5	V	V	V	V
6	Subjek 6	V	V	V	V
7	Subjek 7	V	V	V	V
8	Subjek 8	V	V	V	V
9	Subjek 9	V	V	V	V
10	Subjek 10	V	V	V	V

Tabel 4.2 dan 4.3 menunjukkan hasil pengujian model pengenalan wajah dengan kondisi pencahayaan sedang (40 lux – 100 lux). Pada pengujian ini setiap partisipan melakukan pengujian di lokasi yang sama dengan perangkat yang sama. Dapat dilihat bahwa dalam kondisi pencahayaan sedang dan kamera resolusi tinggi maupun rendah sebagian besar partisipan dapat dikenali. Oleh karena itu dengan kondisi cahaya sedang, baik kamera resolusi tinggi maupun rendah dapat mengenali partisipan dengan baik dan memiliki tingkat keberhasilan cukup tinggi, yaitu 90%. Juga dapat melakukan spoofing detection dengan sangat baik yaitu memiliki akurasi 100%

Tabel 4.2 Hasil Pengujian Pengenalan Wajah Pencahayaan Sedang

No	Subjek	*Spoofing Detection*		Pengenalan Wajah	
		Kamera Resolusi Tinggi	Kamera Resolusi Rendah	Kamera Resolusi Tinggi	Kamera Resolusi Rendah
1	Subjek 1	V	V	V	V
2	Subjek 2	V	V	X	X

Tabel 4.3 Hasil Pengujian Pengenalan Wajah Pencahayaan Sedang (Lanjutan)

No	Subjek	*Spoofing Detection*		Pengenalan Wajah	
		Kamera Resolusi Tinggi	Kamera Resolusi Rendah	Kamera Resolusi Tinggi	Kamera Resolusi Rendah
3	Subjek 3	V	V	V	V
4	Subjek 4	V	V	V	V
5	Subjek 5	V	V	V	V
6	Subjek 6	V	V	V	V
7	Subjek 7	V	V	V	V
8	Subjek 8	V	V	V	V
9	Subjek 9	V	V	V	V
10	Subjek 10	V	V	V	V

Tabel 4.4 dan 4.5 menunjukkan hasil pengujian model pengenalan wajah dengan kondisi pencahayaan rendah (di bawah 10 lux). Pada pengujian ini setiap partisipan melakukan pengujian di lokasi yang sama dengan perangkat yang sama. Dapat dilihat bahwa dalam kondisi pencahayaan rendah dan kamera resolusi tinggi maupun rendah kurang baik. Dengan kondisi pencahayaan rendah keduanya memiliki tingkat keberhasilan yang kurang baik, di mana kamera dengan resolusi tinggi memiliki tingkat keberhasilan sebesar 70% dan kamera dengan resolusi rendah memiliki tingkat keberhasilan sebesar 50%. Juga dapat melakukan spoofing detection dengan sangat baik yaitu memiliki akurasi 100%

Tabel 4.4 Hasil Pengujian Pengenalan Wajah Pencahayaan Rendah

No	Subjek	*Spoofing Detection*		Pengenalan Wajah	
		Kamera Resolusi Tinggi	Kamera Resolusi Rendah	Kamera Resolusi Tinggi	Kamera Resolusi Rendah
1	Subjek 1	V	V	V	V
2	Subjek 2	V	V	X	X
3	Subjek 3	V	V	V	V
4	Subjek 4	V	V	X	X

Tabel 4.5 Hasil Pengujian Pengenalan Wajah Pencahayaan Rendah (Lanjutan)

No	Subjek	*Spoofing Detection*		Pengenalan Wajah	
		Kamera Resolusi Tinggi	Kamera Resolusi Rendah	Kamera Resolusi Tinggi	Kamera Resolusi Rendah
5	Subjek 5	V	V	V	X
6	Subjek 6	V	V	V	V
7	Subjek 7	V	V	V	X
8	Subjek 8	V	V	V	V
9	Subjek 9	V	V	X	X
10	Subjek 10	V	V	V	V

Berdasarkan pengujian pengenalan wajah yang telah dilakukan, dapat disimpulkan bahwa pencahayaan cukup mempengaruhi model dalam mengenali wajah partisipan. Hal ini dapat dilihat dengan menurunnya tingkat keberhasilan dari 100% saat cahaya terang menjadi 90% saat cahaya sedang dan 60% saat cahaya rendah. Sedangkan resolusi kamera cukup mempengaruhi model dalam mengenali wajah partisipan pada kondisi pencahayaan rendah, hal ini dapat dilihat dengan tingkat keberhasilan yang cukup berbeda pada kondisi pencahayaan kurang, di mana kamera dengan resolusi tinggi memiliki tingkat keberhasilan sebesar 70% sedangkan kamera dengan resolusi rendah memiliki tingkat keberhasilan sebesar 50%. Pendeteksian wajah dan spoofing detection dapat dilakukan dengan sangat baik di berbagai kondisi cahaya dan resolusi kamera dengan tingkat akurasi 100%

Berkurangnya akurasi model pengenalan wajah akibat kondisi cahaya dapat diatasi dengan tambahan fitur yang dapat membantu model dalam mengenali wajah pengguna. Salah satu solusi yang dapat diterapkan selanjutnya adalah dengan secara otomatis menaikkan *brightness* layar perangkat ketika akan mengambil foto pengguna untuk diproses oleh model pengenalan wajah tersebut. Dengan demikian, cahaya dari layar akan menerangi wajah pengguna dan memperjelas gambar wajah yang diambil.

4.3.3 Hasil Pengujian Kepuasan Pengguna

Pengujian yang terakhir adalah pengujian kepuasan pengguna. Pengujian kepuasan pengguna dilakukan dengan mengisi kuesioner melalui Google Form. Setiap partisipan akan diminta untuk mencoba menggunakan seluruh fitur pada aplikasi dan memberikan tanggapan dengan mengisi kuesioner. Tabel 4.6 menunjukkan hasil dari kuesioner yang telah diisi oleh sepuluh partisipan.

Tabel 4.6 Hasil Kuisioner Partisipan

No	Pernyataan	Sangat Tidak Setuju	Tidak Setuju	Biasa	Setuju	Sangat Setuju
1.	Tampilan halaman login menarik				30%	70%
2.	Tampilan halaman home menarik			10%	30%	60%
3.	Tampilan halaman absen menarik			10%	20%	70%
4.	Tampilan halaman riwayat menarik				30%	70%
5.	Tampilan halaman ganti password menarik			10%	60%	30%
6.	Proses login mudah dilakukan				20%	80%
7.	Proses pendaftaran wajah mudah dilakukan			10%	20%	70%
8.	Proses absensi mudah dilakukan				30%	70%
9.	Proses ganti password mudah dilakukan				20%	80%
10.	Fitur riwayat absensi sangat membantu dalam melacak jejak absensi				30%	70%
11.	Fitur absensi sangat membantu dalam memudahkan pendataan absensi				20%	80%
12.	Fitur ganti password sangat membantu				20%	80%
13.	Aplikasi mudah untuk digunakan				20%	80%

Melalui hasil kuesioner tersebut dapat dilihat bahwa tampilan aplikasi sudah cukup bagus dengan rata-rata 60% partisipan menjawab “Sangat Setuju” untuk pertanyaan terkait tampilan aplikasi. Juga dengan rata-rata 76,25% partisipan menjawab “Sangat Setuju” untuk pertanyaan terkait kenyamanan dan fitur dalam aplikasi, dapat disimpulkan bahwa aplikasi dapat digunakan dengan cukup lancar oleh pengguna dan sesuai dengan fungsinya untuk membantu pengguna dalam proses presensi sehari-hari.

Terdapat beberapa kritik dan saran yang disampaikan oleh partisipan. Kritik dan saran terkait dengan proses presensi adalah tulisan yang perlu diperbesar agar petunjuk dapat dibaca dari jarak pengguna dengan perangkat ketika melakukan proses presensi. Juga beberapa partisipan mengalami kesulitan untuk mengetahui kapan harus menghadap ke depan lagi setelah mengikuti perintah menghadap ke arah lain. Selain itu terdapat tanggapan terkait dengan *spoofing detection*, saat ini aplikasi akan mengacak empat macam perintah, yaitu menghadap kiri, kanan, atas, dan bawah. Tetapi beberapa partisipan merasa kurang nyaman dan sedikit kesulitan ketika melakukan hadap atas dan bawah.

Menanggapi kritik dan saran yang disampaikan, peneliti berikutnya dapat memperbesar tulisan pada halaman presensi dan juga menambahkan tanda yang memberitahu pengguna untuk kembali menghadap ke depan setelah melakukan *spoofing detection*, tanda dapat berupa suara atau getaran yang dapat diketahui pengguna tanpa perlu melihat ke arah layar. Selain itu juga untuk mengurangi jumlah perintah yang diacak oleh aplikasi dari empat macam perintah menjadi dua macam perintah, yaitu menghadap kiri dan kanan saja.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari pengembangan aplikasi *multiplatform* presensi dengan *face recognition* menggunakan FaceNet, dapat disimpulkan bahwa aplikasi yang telah berhasil dikembangkan memiliki tingkat akurasi keberhasilan mengenali wajah sebesar 83.3%. Di mana kondisi pencahayaan cukup mempengaruhi tingkat keberhasilan mengenali wajah, dengan tingkat keberhasilan paling tinggi pada kondisi pencahayaan terang, yaitu sebesar 100% untuk kamera resolusi tinggi dan resolusi rendah. Sedangkan resolusi kamera cukup mempengaruhi akurasi pada kondisi pencahayaan rendah, hal ini dapat dilihat dari tingkat keberhasilan yang berbeda pada kondisi pencahayaan rendah, di mana kamera resolusi tinggi memiliki keberhasilan sebesar 70% sedangkan kamera resolusi rendah memiliki keberhasilan sebesar 50%. Aplikasi dapat melakukan *spoofing detection* dengan baik di setiap kondisi pencahayaan dan resolusi kamera dengan tingkat akurasi untuk *spoofing detection* sebesar 100%. Aplikasi yang dikembangkan juga memiliki tampilan menarik dan mudah digunakan, dengan rata-rata 60% partisipan sangat setuju tampilan aplikasi menarik dan rata-rata 76.25% partisipan setuju aplikasi mudah untuk digunakan.

Pendeteksi lokasi pada aplikasi yang dikembangkan tidak dapat mendeteksi penggunaan *Fake GPS* ketika aplikasi dijalankan. Sehingga aplikasi menerima lokasi yang salah, sesuai dengan lokasi yang dipilih pada aplikasi *Fake GPS*, dan tidak mendeteksi lokasi pengguna yang sebenarnya. Aplikasi yang dikembangkan juga tidak dapat mendeteksi kecurangan melakukan presensi menggunakan *video call*. Ketika melakukan presensi dengan melalui *video call*, wajah dapat terdeteksi dan dikenali dengan lancar.

5.2 Saran

Saran untuk pengembangan aplikasi *multiplatform* dengan *face recognition* adalah untuk menambahkan fitur untuk menaikkan level *brightness* layar secara otomatis ketika hendak mengambil gambar wajah pengguna sehingga cahaya dari

layar dapat menerangi wajah pengguna dan memberikan gambar wajah pengguna yang lebih jelas untuk diproses. Saran berikutnya adalah untuk menambahkan pendeteksi objek yang dapat mendeteksi perangkat pengguna, hal ini agar mencegah pengguna melakukan kecurangan melalui *video call*. Dan untuk mencegah penggunaan *fake GPS*, dapat ditambahkan pemeriksaan apakah pengguna menyalakan *mock location* ketika sedang melakukan presensi.

DAFTAR PUSTAKA

Chaf, M. B., Hultberg, A., & Yams, N. B. (2022). Post-Pandemic Office Work: Perceived Challenges and Opportunities for a Sustainable Work Environment. *Sustainability*.

Kumaran, I., Firmansyah, M. R., Fauziah, E., Hutahaean, Y., Suryana, A., Sidik, A. D., Kusumah, I. H. (2022). Pengenalan Wajah Menggunakan PendekatanBerbasis Pengukuran dan Metode Segmentasi dalamBerbagai Posisi dan Pencahayaan. *Jurnal Teknik Elektro*, 5-8.

Atmaja, N. S. (2022). Implementasi Metode Levensthein Distance dan Cosine Similarity untuk Deteksi Kemiripan Gambar. *Riau Journal of Computer Science*, 85-93.

Basurah, M., Swastika, W., & Kelana, O. H. (2019). Implementation of Face Recognition and Liveness Detection System Using TensorFlow.js. *Kinetik*, 197-206.

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*. Long Beach.

Buana, K. S. (2021). Penerapan Pengenalan Wajah Untuk Aplikasi Absensi dengan Metode. *Media Informatika Budidarma*, 1008-1017.

Evelyn, Adipranata, R., & Gunadi, K. (2022). Sistem Presensi Mahasiswa Menggunakan Face. *Jurnal Infra*.

GeeksforGeeks. (2021, July 14). *What are Web Services?* Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/what-are-web-services/?ref=gcse

Google. (2023, March 20). *Flutter Architectural Overview.* Retrieved from Flutter Documentation: https://docs.flutter.dev/resources/architectural-overview

Iskandar, Abdurahman, U. T., & Abdurahman, U. T. (2022). Rancang Bangun Aplikasi Kehadiran Siswa Menggunakan. *Rancang Bangun Aplikasi Kehadiran Siswa Menggunakan*, 284-295.

Putra, S., Fitri, I., & Fitri, I. (2021). Absensi Pengenalan Wajah Menggunakan Menggunakan Algoritma. *Journal of Applied Informatics and Computing*, 21-27.

Praba, A. D., & Safitri, M. (2020). Studi Perbandingan Performansi Antara MySQL dan PostgreSQL. Jurnal Khatulistiwa Informatika, 88-93.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 815-823). Boston.

TensorFlow. (2023, March 01). *Model conversion overview*. Retrieved from TensorFlow: https://www.tensorflow.org/lite/models/convert

Aprilia, P., 2021. *Apa itu Java? Pengertian, Kelebihan, Kekurangan, dan Contohnya*. Retrieved from NiagaHoster: https://www.niagahoster.co.id/blog/java-adalah/