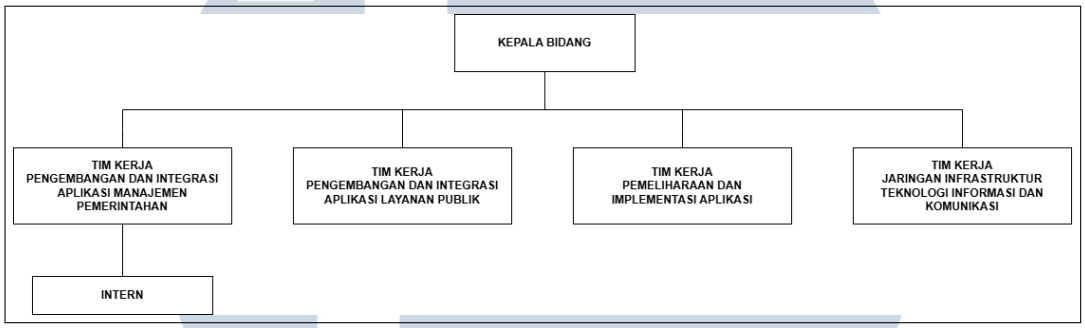
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1. Struktur Tim Proyek

Selama menjalani program magang di Dinas Komunikasi dan Informatika (Diskominfo) Kabupaten Tangerang, penempatan dilakukan di Bidang Pengelolaan Informatika yang berfokus pada pengembangan aplikasi digital pemerintah daerah. Tugas utama yang diberikan adalah merancang sistem *machine learning* berbasis *face recognition* untuk verifikasi absensi pegawai. Dalam pelaksanaannya, koordinasi dilakukan secara langsung dengan pegawai pengembang sistem dan didampingi oleh pembimbing lapangan, Bapak Budi Kurniawan, yang memberikan arahan serta evaluasi terhadap hasil pekerjaan. Koordinasi berlangsung melalui pertemuan tatap muka di kantor dan juga didukung oleh media komunikasi seperti *WhatsApp* dan *Zoom Meeting*. Pembagian tugas serta pelaporan perkembangan proyek dilakukan dalam sesi diskusi mingguan. Peran yang dijalankan mencakup pengembangan awal sistem, mulai dari proses pengumpulan data hingga penerapan model *face recognition*, serta kontribusi dalam integrasi sistem ke dalam infrastruktur teknologi yang telah tersedia di Diskominfo.

3.2 Tugas yang Dilakukan

Selama kegiatan magang di Dinas Komunikasi dan Informatika Kabupaten Tangerang, terdapat beberapa kontribusi yang diberikan dalam pengembangan sistem *machine learning* berbasis *face recognition*. Adapun rincian tugas-tugas yang dilakukan adalah sebagai berikut.

1. Mengembangkan sistem absensi berbasis face recognition, termasuk membuat *flowchart* sistem.
2. Membangun *backend* API menggunakan *Python* dan *framework Flask*.
3. Mengimplementasikan model FaceNet untuk mengekstrak *face embedding* dari gambar wajah.
4. Membuat *endpoint /predict* untuk proses verifikasi wajah menggunakan metode *cosine similarity*.
5. Mengembangkan *endpoint /register* khusus untuk admin, dilengkapi validasi JWT agar hanya admin yang bisa mengakses fitur pendaftaran pengguna baru.
6. Menambahkan validasi form input seperti NIP, nama, dan jumlah gambar yang wajib 5 file, serta memastikan ekstensi file adalah jpg, jpeg, atau png.
7. Mengintegrasikan pemeriksaan isi file gambar menggunakan untuk memastikan file yang diunggah adalah gambar asli dan tidak rusak.
8. Mengimplementasikan sistem login menggunakan JWT, termasuk pengaturan durasi token dan hashing password untuk keamanan.
9. Melakukan pengujian sistem menggunakan dataset LFW, untuk mengevaluasi akurasi model FaceNet.

3.3 Uraian Pelaksanaan Magang

Berikut adalah uraian kegiatan magang yang telah dilaksanakan di Dinas Komunikasi dan Informatika (Diskominfo) Kabupaten Tangerang, sebagaimana dirangkum dalam Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke-	Pekerjaan yang dilakukan
1	Merancang sistem absensi berbasis face recognition untuk verifikasi absen.
2	Membuat *flowchart* sistem untuk menggambarkan alur kerja proses absensi otomatis.

Minggu Ke-	Pekerjaan yang dilakukan
3	Menentukan *framework* yang digunakan, yaitu *Python* dan *Flask* untuk membangun *backend* API. Menyusun struktur proyek yang modular agar pengembangan lebih terorganisir dan mudah dipelihara.
4	Mengimplementasikan model FaceNet untuk mengekstrak *face embedding* dari gambar wajah. Menyimpan embedding wajah dan data pengguna (nama & NIP) ke dalam file .pkl dan .txt saat registrasi.
5	Membuat *endpoint /predic*t untuk proses verifikasi wajah berdasarkan gambar yang dikirim. Membuat sistem perbandingan embedding wajah menggunakan cosine similarity untuk menentukan identitas.
6	Menyusun respons API untuk mengembalikan hasil prediksi (NIP dan nama) jika wajah terverifikasi. Melakukan validasi input file gambar dan pembersihan file temporary setelah diproses. Menguji keberhasilan proses verifikasi wajah dan confidence level pada log server.
7	Mengimplementasikan e*ndpoint /register* untuk admin guna mendaftarkan pengguna baru beserta data wajah. Menambahkan validasi token JWT untuk membatasi akses hanya bagi role *admin*.
8	Membuat validasi form nip, name, dan jumlah file gambar tepat 5 foto. Menerapkan validasi ekstensi file gambar agar hanya menerima jpg, jpeg, dan png.
9	Mengintegrasikan validasi isi file menggunakan library PIL untuk memastikan gambar asli dan tidak *corrupt*.
10	Menghubungkan proses *register* dengan fungsi pembentukan *face embedding* menggunakan model FaceNet. Menyimpan *embedding* wajah dalam file .pkl dan data pengguna dalam file .txt.
11	Menyusun *endpoint /login* untuk autentikasi user menggunakan JWT. Mengatur konfigurasi JWT dengan *secret key* dan durasi token selama 5 menit.

Minggu Ke-	Pekerjaan yang dilakukan
12	Menambahkan *payload role* dalam JWT untuk otorisasi akses *endpoint* tertentu. Membuat fungsi verifikasi password menggunakan hashing.
13	Mendesain respons API login yang menampilkan token jika berhasil atau *error* jika gagal. Menerapkan fungsi untuk validasi file sebelum diproses oleh sistem *embedding*.
14	Menggunakan dataset LFW untuk menguji akurasi sistem *face recognition*. Melakukan pengujian model FaceNet terhadap variasi gambar wajah dari dataset LFW.

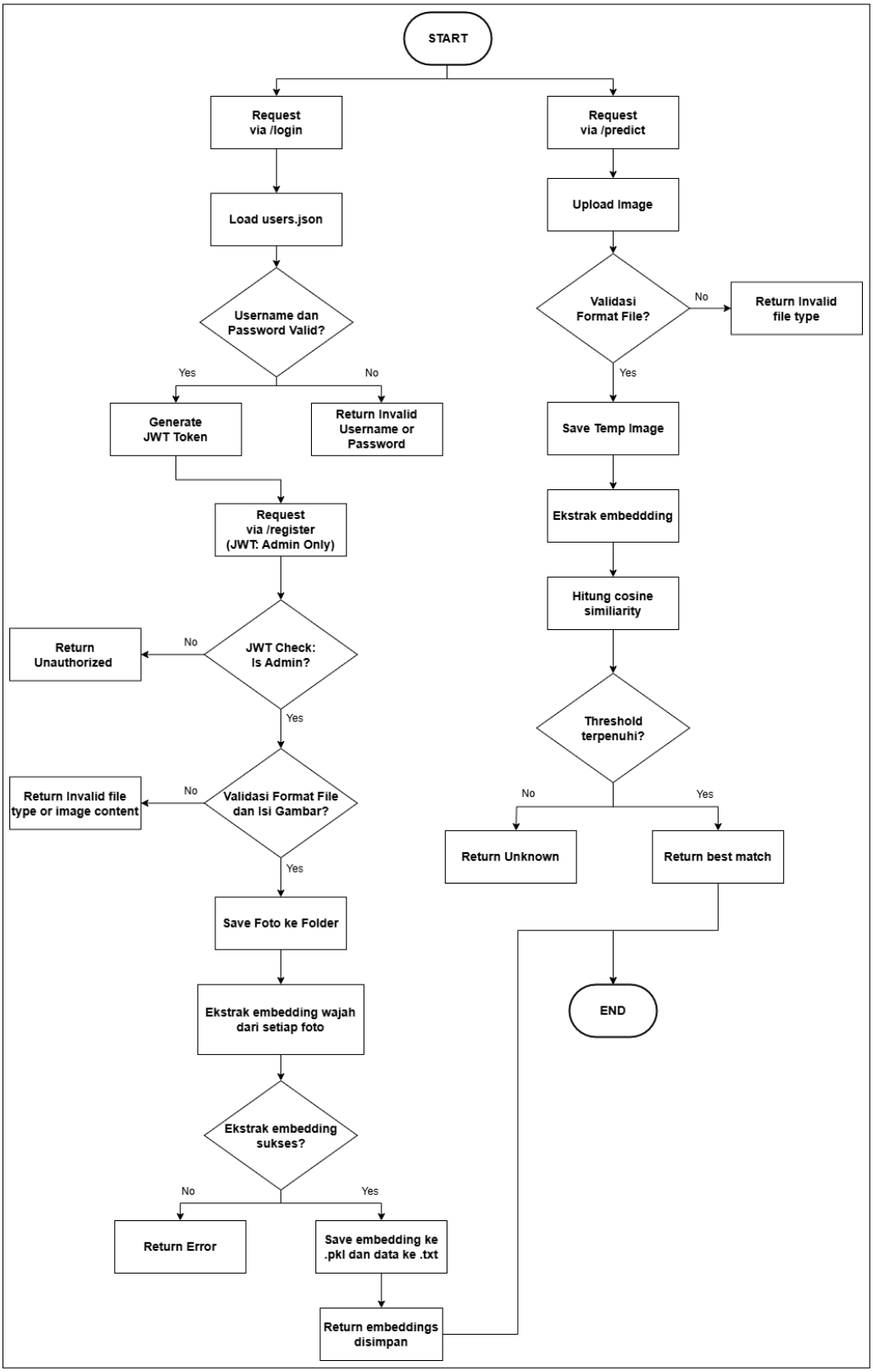
3.3.1 Perancangan

A Uraian Masalah dan Kebutuhan

Sebelum membuat sistem absensi berbasis *face recognition*, absensi di Dinas Komunikasi dan Informatika (Diskominfo) Kabupaten Tangerang masih belum dilengkapi dengan proses verifikasi otomatis. Pengguna mengirimkan foto mereka ke sistem tanpa proses verifikasi isi foto tersebut. Foto yang dikirimkan oleh pengguna ke sistem diterima begitu saja tanpa ada proses pengecekan lebih lanjut. Hal ini membuka potensi penyalahgunaan, seperti penggunaan foto orang lain, hasil tangkapan layar, atau gambar hasil rekayasa digital [12]. Tidak adanya proses verifikasi otomatis ini menjadi celah serius yang dapat dimanfaatkan oleh oknum untuk melakukan kecurangan. Melihat kendala tersebut, pihak pengelola sistem memutuskan untuk menginisiasi pengembangan sebuah sistem absensi otomatis berbasis *face recognition*. Sistem ini dirancang untuk mencocokkan foto yang dikirim oleh pengguna dengan data wajah yang telah direkam dan disimpan sebelumnya saat proses registrasi. Dengan demikian, hanya pengguna dengan wajah yang sesuai dan terverifikasi yang dapat dianggap hadir.

B Gambaran Umum Sistem

Sistem dirancang untuk melakukan proses absensi secara otomatis menggunakan teknologi *face recognition*. Untuk memberikan gambaran umum mengenai sistem yang dirancang, berikut adalah diagram alur (*flowchart*) sistem absensi berbasis *face recognition*.



Gambar 3.2. Flowchart Sistem

Sistem bekerja dengan cara mendeteksi wajah dari gambar yang dikirim, kemudian mengekstrak fitur wajah tersebut menggunakan model FaceNet [4]. Fitur yang telah diekstrak ini akan dibandingkan dengan data *embedding* wajah yang telah disimpan dalam basis data saat proses registrasi. Proses pencocokan dilakukan dengan menghitung tingkat kemiripan *(cosine similarity)* antara dua vektor wajah [13]. Jika tingkat kemiripan memenuhi ambang batas *(threshold)* tertentu, maka absensi dianggap valid dan pengguna dinyatakan hadir. Dengan diterapkannya sistem ini, diharapkan proses verifikasi kehadiran menjadi lebih aman, cepat, dan akurat, serta mengurangi potensi penyalahgunaan data absensi.

C Framework yang digunakan

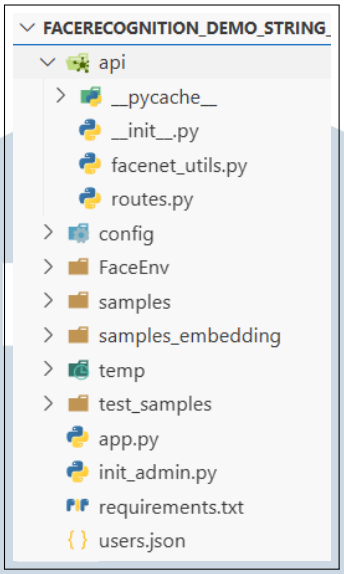
Aplikasi ini dirancang menggunakan bahasa pemrograman *Python* untuk membangun sistem *backend* yang menangani proses otentikasi pengguna, pendaftaran data wajah, serta verifikasi wajah berbasis *machine learning*. *Python* dipilih karena memiliki banyak *library* yang mendukung pengembangan sistem *machine learning*. Bahasa ini juga sesuai dengan kebutuhan teknis *developer* dalam merancang dan mengimplementasikan proses pengenalan wajah secara otomatis menggunakan model *machine learning*. Selain itu, *Python* mudah dibaca dan dipahami, sehingga mempercepat proses pengembangan dan pemeliharaan aplikasi.

Sistem *backend* dibuat menggunakan *Flask*, yaitu *framework* ringan berbasis *Python* yang mudah digunakan dan fleksibel. *Flask* digunakan untuk membuat REST API yang menghubungkan tampilan aplikasi dengan proses yang berjalan di server. Dengan menggunakan fitur *blueprint*, pengelolaan route API seperti /login, /register, dan /predict menjadi lebih rapi. Dan untuk keamanan menggunakan *library flask_jwt_extended* agar hanya pengguna dengan role 'admin' yang bisa mengakses route API /register dan mendaftarkan data wajah menggunakan sistem token JWT (JSON Web Token).

3.3.2 Implementasi

A Struktur Proyek

Struktur proyek aplikasi verifikasi wajah ini terdiri dari beberapa folder dan file utama yang memiliki peran penting dalam pengembangan sistem. Pengorganisasian struktur folder dibuat agar kode mudah dipahami dan dikelola. Berikut adalah tangkapan layar dari struktur proyek.



Gambar 3.3. Struktur Proyek

Berikut merupakan penjelasan dari komponen utama dalam proyek.

1. facenet_utils.py: File ini berisi fungsi untuk pengolahan wajah, seperti memuat model FaceNet, mengekstrak embedding (fitur wajah), serta menghitung jarak antar wajah.

2. routes.py: Berisikan endpoint REST API, seperti /login, /register, dan /predict, yang digunakan untuk proses autentikasi dan verifikasi wajah.

3. app.py: Merupakan file yang menjalankan aplikasi Flask. Di dalamnya terdapat konfigurasi seperti pengaturan JWT, inisialisasi server, dan registrasi blueprint API

4. users.json: Merupakan file penyimpanan data pengguna dalam format JSON. File ini menyimpan username, password yang sudah di-hash, serta role dari pengguna.

B Implementasi FaceNet

FaceNet adalah model *machine learning* yang telah melalui proses *pre-trained* dan secara luas digunakan untuk mengekstraksi fitur wajah dalam bentuk representasi numerik, yang disebut sebagai *face embedding* [4]. Karena merupakan model *pre-trained*, FaceNet dapat langsung digunakan tanpa memerlukan proses

training dari awal, sehingga mempercepat pengembangan sistem dan mengurangi kebutuhan komputasi secara signifikan. Hal ini menjadikannya pilihan ideal untuk implementasi verifikasi wajah di berbagai aplikasi, termasuk dalam proyek ini. *Embedding* yang dihasilkan oleh FaceNet bersifat unik untuk setiap individu, dan dapat digunakan sebagai identitas digital yang stabil dalam proses pencocokan dan verifikasi wajah. Dalam sistem ini, setelah pengguna dan foto dinyatakan valid pada tahap register data, setiap gambar wajah akan diproses melalui Kode 3.1 berikut.

```
facenet = FaceNet()
model = facenet.model

def get_embedding(image_path):
    try:
        face = preprocess_image(image_path)
        return model.predict(face)[0]
    except Exception as e:
        print(f"Error extracting embedding: {e}")
        return None
```

Kode 3.1: Potongan kode implementasi FaceNet

Fungsi *get_embedding()* mengambil path gambar yang telah diproses sebelumnya, kemudian mengirimnya ke model FaceNet untuk diubah menjadi embedding. Hasil embedding ini akan disimpan dalam file berformat .pkl menggunakan *library pickle*, yang kemudian digunakan pada proses pencocokan (matching) wajah saat verifikasi. Selain itu, informasi pengguna seperti nama dan NIP juga disimpan dalam file teks (.txt). Implementasi FaceNet ini memberikan keakuratan tinggi dalam mengenali wajah, bahkan ketika terdapat variasi pencahayaan, pose, atau ekspresi. Dengan menggunakan representasi *embedding*, sistem dapat membandingkan wajah yang berbeda secara efisien berdasarkan jarak vektor embedding-nya.

C Implementasi Fitur Predict (Verifikasi Wajah)

Fitur predict atau verifikasi wajah digunakan untuk mengenali identitas seseorang berdasarkan gambar wajah yang dikirimkan. Endpoint ini bisa digunakan untuk pengguna dengan *role admin* dan *user*. Cara kerjanya adalah dengan mencocokkan gambar yang diterima dengan data wajah yang sudah terdaftar sebelumnya. Proses ini menjadi inti dari sistem verifikasi berbasis wajah yang dibangun. Berikut adalah *Python script* yang digunakan untuk *route /predict*, dapat

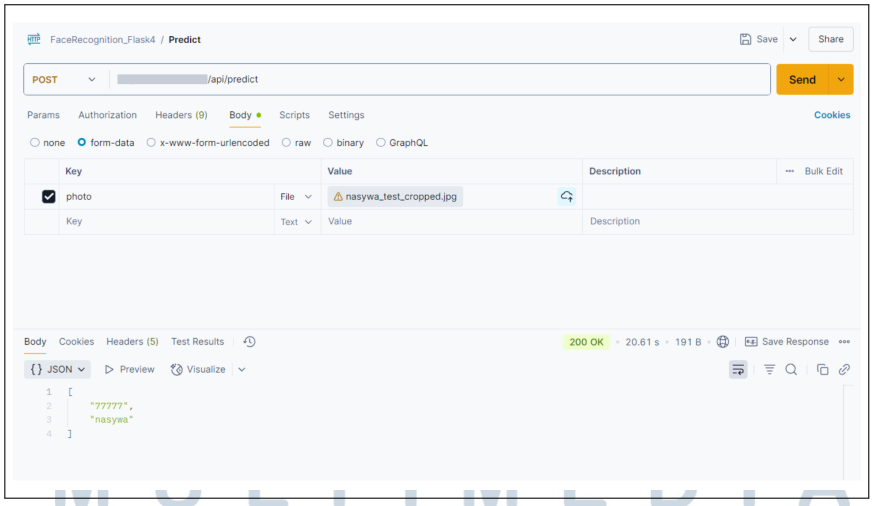
dilihat pada Kode 3.2.

```
@api_blueprint.route('/predict', methods=['POST'])
def predict():
    if 'photo' not in request.files:
        return jsonify({"error": "Image file is required"}), 400

    photo = request.files['photo']

    if not allowed_file(photo.filename):
        return jsonify({"error": "File type not allowed. Only jpg, jpeg, png accepted."}), 400

    temp_filename = f"{uuid4().hex}.jpg"
    image_path = os.path.join(TEMP_DIR, temp_filename)
    photo.save(image_path)

    test_embedding = get_embedding(image_path)
    os.remove(image_path)

    if test_embedding is None:
        return jsonify({"error": "Failed to extract face embedding"}), 500

    best_nip = "unknown"
    best_name = "unknown"
    best_similarity = 0.0

    for folder_name in os.listdir(EMBEDDINGS_DIR):
        pkl_path = os.path.join(EMBEDDINGS_DIR, folder_name, f"{folder_name}.pkl")
        txt_path = os.path.join(EMBEDDINGS_DIR, folder_name, f"{folder_name}.txt")

        if not os.path.exists(pkl_path) or not os.path.exists(txt_path):
            continue

        with open(pkl_path, "rb") as f:
            known_embeddings = pickle.load(f)

        similarity = compare_faces(test_embedding, known_embeddings, THRESHOLD)

```

```
37          if similarity > best_similarity:
38              best_similarity = similarity
39              best_nip, best_name = read_txt(txt_path)
40
41      print(f"Best match: {best_nip} (Confidence: {round(
        best_similarity * 100, 2)}%)")
42      return jsonify(best_nip, best_name), 200
```

Kode 3.2: Potongan kode route /predict

Pada implementasi ini, pengguna akan mengunggah satu file gambar melalui *request* POST. Server terlebih dahulu memvalidasi file dan formatnya (hanya menerima jpg, jpeg, atau png). Gambar tersebut kemudian disimpan di direktori sementara sebelum diproses. Fungsi *get_embedding()* akan digunakan untuk menghasilkan embedding wajah dari gambar tersebut menggunakan model FaceNet. Jika tidak ditemukan wajah dalam gambar atau embedding gagal diproses, sistem akan mengembalikan respons error. Selanjutnya, sistem akan membandingkan *embedding* hasil input dengan seluruh data *embedding* yang telah disimpan pada *database* lokal. Perbandingan dilakukan dengan membaca setiap file .pkl (yang berisi *embedding* pengguna yang terdaftar) dan menggunakan fungsi *compare_faces()* untuk menghitung skor kemiripan. Berikut adalah respons yang akan diberikan ketika proses prediksi sukses.



Gambar 3.4. Respons API Predict

Apabila skor kemiripan *cosine similarity* melebihi ambang batas *(threshold)* dan merupakan nilai tertinggi sejauh ini, maka informasi NIP dan nama pengguna dari file .txt yang bersesuaian akan disimpan sebagai hasil terbaik. Akhirnya, sistem

akan mengembalikan respons berupa NIP dan nama dari pengguna yang paling mirip dengan gambar yang diunggah, bersama dengan tingkat kepercayaan yang dicetak ke log untuk keperluan debugging.

D Implementasi Fitur Register (Pendaftaran Data Wajah)

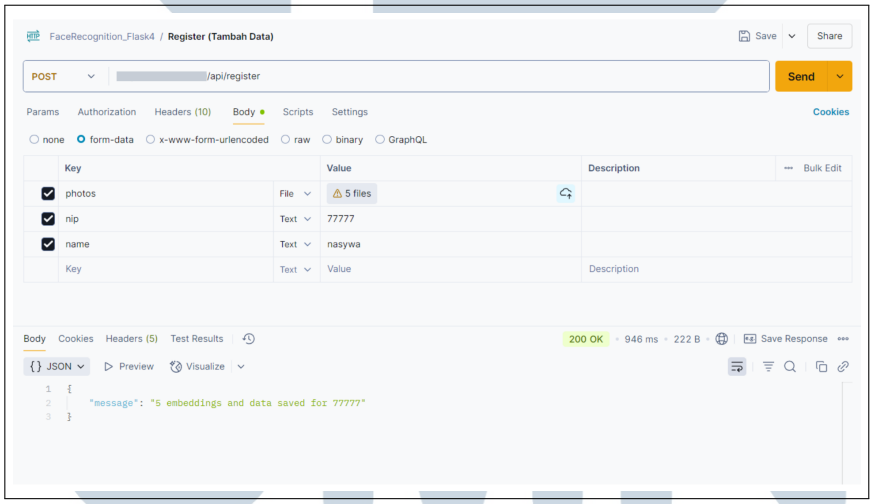
Fitur ini digunakan oleh admin untuk menambahkan informasi pengguna baru ke sistem beserta data wajah mereka. Tujuannya adalah agar sistem dapat mengenali pengguna berdasarkan wajah saat proses verifikasi. *Endpoint* ini hanya dapat diakses oleh pengguna yang memiliki role admin, yang diverifikasi melalui token JWT. Berikut adalah *Python script* yang digunakan untuk *route /register*, dapat dilihat pada Kode 3.3.

```
@api_blueprint.route('/register', methods=['POST'])
@jwt_required()
def register():
    claims = get_jwt()
    if claims.get("role") != "admin":
        return jsonify({"error": "Unauthorized. Only admin can register users."}), 403

    if 'nip' not in request.form:
        return jsonify({"error": "Person nip is required"}), 400
    elif 'name' not in request.form:
        return jsonify({"error": "Person name is required"}), 400

    person_nip = request.form['nip']
    person_name = request.form['name']
    files = request.files.getlist('photos')

    if len(files) < 5:
        return jsonify({"error": "Exactly 5 images are required"}), 400

    for file in files:
        if not allowed_file(file.filename):
            return jsonify({"error": f"Invalid file type: {file.filename}. Only jpg, jpeg, png allowed."}), 400

    for file in files:
        if not allowed_file(file.filename):
```

```
            return jsonify({"error": f"Invalid file type: {file.filename}. Only jpg, jpeg, png allowed."}), 400
        if not is_valid_image(file):
            return jsonify({"error": f"Invalid image content: {file.filename} is not a real image"}), 400

    result, status = register_user_and_generate_embeddings(person_nip, person_name, files)
    return jsonify(result), status
```

Kode 3.3: Potongan kode route /register

Dalam proses ini, admin perlu mengisi NIP (Nomor Induk Kependudukan), nama, dan mengunggah tepat lima foto wajah. Sistem akan memeriksa apakah pengguna yang mengakses *endpoint* ini memiliki hak sebagai admin. Jika bukan admin, maka akses akan ditolak. Dan jika seluruh validasi berhasil, maka file gambar akan diteruskan ke proses ekstraksi fitur wajah menggunakan model FaceNet. Berikut adalah contoh respons ketika proses registrasi data baru berhasil.



Gambar 3.5. Respons API Register

Hasil ekstraksi gambar adalah berupa *face embedding* yang disimpan dalam file pickle (.pkl). Sementara informasi pengguna seperti nama dan NIP akan disimpan dalam bentuk file text (.txt). Seluruh data pengguna yang berhasil diregistrasi tersebut dicatat ke dalam database lokal untuk mempermudah pengelolaan dan integrasi sistem ke depannya. Fitur ini memastikan bahwa hanya admin yang bisa menambahkan pengguna, dan setiap wajah yang didaftarkan memiliki kualitas yang baik agar bisa dikenali dengan akurat oleh sistem saat verifikasi dilakukan.

E Implementasi Keamanan Sistem

Dalam membangun sistem verifikasi wajah berbasis *Flask*, aspek keamanan menjadi prioritas utama untuk mencegah penyalahgunaan akses maupun manipulasi data [14]. Oleh karena itu, dua pendekatan utama diterapkan dalam sistem ini yaitu, penggunaan *JSON Web Token (JWT)* dalam proses login untuk mengatur autentikasi dan otorisasi pengguna terhadap fitur-fitur tertentu dalam sistem serta validasi terhadap tipe dan isi file gambar yang diunggah, guna memastikan bahwa hanya file yang sah dan sesuai standar yang dapat diproses. Kedua pendekatan ini diimplementasikan langsung di sisi *backend* melalui beberapa endpoint yang telah dirancang secara khusus.

E.1 Penggunaan JWT Token dalam Proses Login

JSON Web Token (JWT) adalah metode autentikasi dan otorisasi berbasis token yang bersifat *stateless* [15]. JWT digunakan untuk menjamin bahwa hanya pengguna yang telah terverifikasi yang dapat mengakses fitur-fitur tertentu dalam sistem. JWT terdiri dari tiga bagian utama: *header, payload*, dan *signature*. Informasi penting seperti identitas pengguna dan hak akses (misalnya sebagai admin) disimpan dalam *payload* dan dikunci menggunakan *signature* yang dibentuk dengan *secret key*. Untuk mengaktifkan penggunaan JWT dalam aplikasi Flask, dilakukan konfigurasi *secret key* dan masa berlaku token. Berikut adalah Python script yang digunakan untuk penggunaan JWT, dapat dilihat pada Kode 3.4.

```
app.config["JWT_SECRET_KEY"] = os.getenv("JWT_SECRET_KEY", "secret-key")
app.config["JWT_ACCESS_TOKEN_EXPIRES"] = timedelta(minutes=5)

jwt = JWTManager(app)
```

Kode 3.4: Potongan kode untuk penggunaan JWT

Konfigurasi tersebut memungkinkan setiap token yang dibuat memiliki batas waktu kadaluwarsa selama 5 menit, serta menjamin keamanan data melalui *secret key* yang bisa diatur melalui variabel lingkungan. Token ini kemudian diberikan kepada user saat berhasil login, dan digunakan kembali untuk mengakses endpoint yang membutuhkan autentikasi.

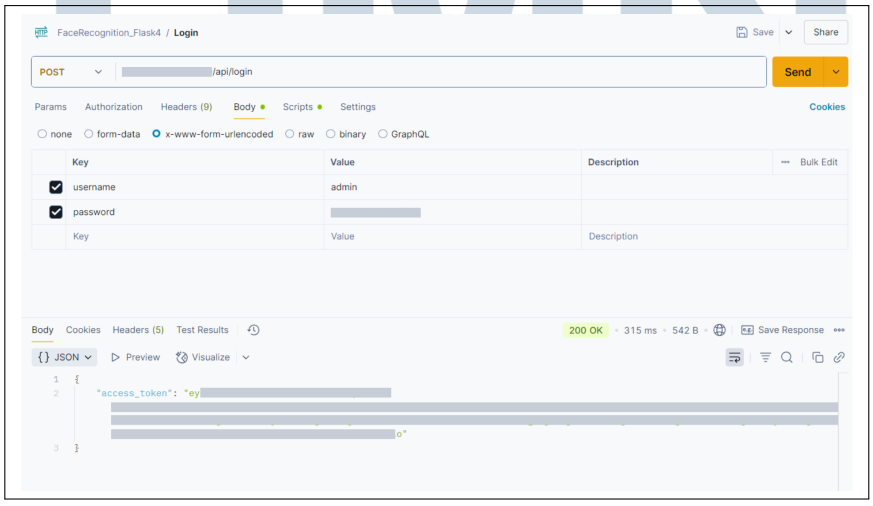
Implementasi JWT dalam proses login dilakukan melalui *endpoint /login*. Ketika pengguna mengirimkan data username dan password ke *endpoint /login*, sistem akan memverifikasi kredensial tersebut dengan mencocokkannya terhadap

data yang tersimpan dalam file users.json. Jika verifikasi berhasil, sistem akan membuat access token menggunakan fungsi *create_access_token()*.

```python
@api_blueprint.route('/login', methods=['POST'])
def login():
    data = request.form
    username = data.get("username")
    password = data.get("password")

    users = load_users()
    user = users.get(username)

    if user and verify_password(password, user["password"]):
        access_token = create_access_token(
            identity=username,
            additional_claims={"role": user["role"]}
        )
        return jsonify(access_token=access_token), 200

    return jsonify({"error": "Invalid username or password"}), 401
```

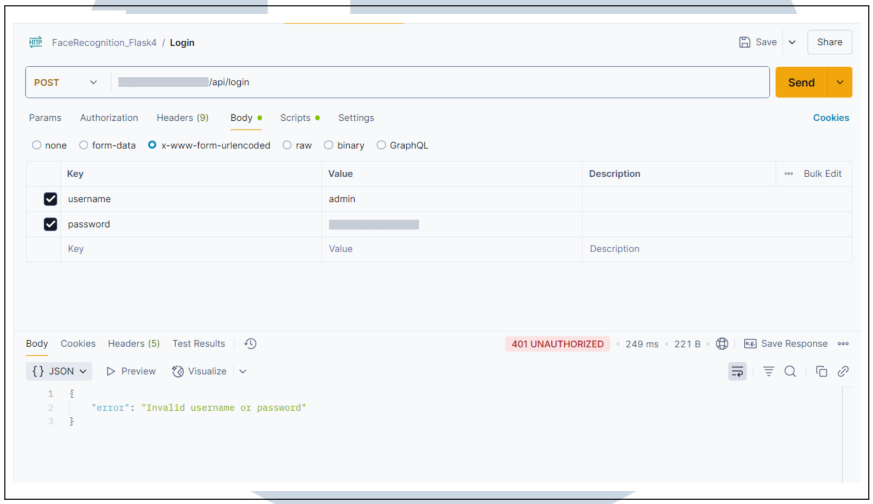
Kode 3.5: Potongan kode route /login

Token JWT yang diterima oleh pengguna akan digunakan untuk mengakses endpoint lain yang membutuhkan autentikasi, seperti register. Token ini dikirim melalui *header Authorization* dalam setiap permintaan. Jika token valid dan menunjukkan bahwa pengguna memiliki peran yang sesuai, maka akses akan diberikan.



Gambar 3.6. Respons API Login Success

Token ini menyimpan identitas pengguna serta informasi tambahan seperti *role 'admin'*. Token yang dihasilkan akan digunakan pada setiap permintaan berikutnya ke *endpoint* yang membutuhkan autentikasi. Sebaliknya, jika pengguna memberikan informasi yang salah (baik username tidak ditemukan atau password tidak cocok), maka sistem akan menolak login dan memberikan response error seperti pada Gambar 3.7 di berikut.



Gambar 3.7. Respons API Login Failed

Dengan implementasi JWT ini, sistem login tidak hanya mengautentikasi identitas pengguna, tetapi juga mengatur hak akses pengguna terhadap endpoint lain. JWT memberi fleksibilitas dan efisiensi tinggi dalam manajemen sesi dan otorisasi tanpa membebani server dengan penyimpanan data sesi pengguna [16].

E.2 Validasi Tipe dan Isi File Gambar

Selain pembatasan akses, sistem juga menerapkan validasi file untuk memastikan bahwa file yang diunggah benar-benar merupakan gambar dalam format yang sesuai dan bukan file berbahaya atau rusak. Berikut adalah *Python script* yang digunakan untuk validasi, dapat dilihat pada Kode 3.6.

```
for file in files:
    if not allowed_file(file.filename):
        return jsonify({"error": f"Invalid file type: {file.filename}. Only jpg, jpeg, png allowed."}), 400
    if not is_valid_image(file):
```

```
        return jsonify({"error": f"Invalid image content: {file.filename} is not a real image"}), 400
```

Kode 3.6: Potongan kode untuk validasi

Sebelum file diproses lebih lanjut, sistem memverifikasi setiap file dengan dua tahapan. Pertama, file diperiksa apakah memiliki format atau ekstensi valid. Kedua, dilakukan pemeriksaan terhadap isi file untuk memastikan file tersebut benar-benar merupakan gambar.

```
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

Kode 3.7: Potongan kode untuk validasi format file

Kode 3.7. ini memeriksa apakah file memiliki format file yang termasuk dalam daftar yang diperbolehkan. Hal ini mencegah pengguna mengunggah file dengan tipe yang tidak didukung atau berpotensi membahayakan sistem.

```
def is_valid_image(file_storage):
    try:
        image = Image.open(file_storage.stream)
        image.verify()
        file_storage.stream.seek(0)
        return True
    except Exception:
        return False
```

Kode 3.8: Potongan kode untuk validasi isi file

Fungsi ini menggunakan *library* PIL untuk membuka dan memverifikasi isi file sebagai gambar. Jika file tidak dapat diverifikasi atau mengalami error saat dibuka sebagai gambar, maka file tersebut dianggap tidak valid dan proses unggah akan dihentikan. Dengan kombinasi validasi tersebut, sistem dapat mencegah unggahan file yang tidak sesuai atau mencurigakan, serta melindungi proses pembentukan embedding wajah agar hanya dijalankan terhadap data yang valid [17].

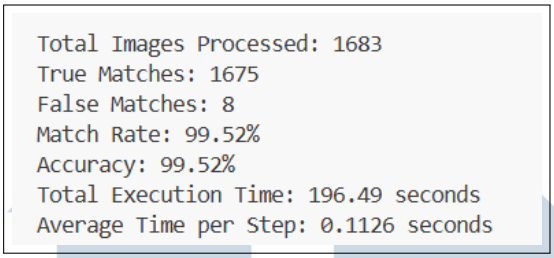
3.3.3 Pengujian

A Dataset

Dalam pengujian sistem face recognition untuk verifikasi absensi, digunakan dataset *Labeled Faces in the Wild* (LFW) sebagai sumber data wajah. LFW dikembangkan oleh peneliti dari *University of Massachusetts, Amherst*, dan berisi 13.233 gambar wajah dari 5.749 individu yang dikumpulkan secara daring [18]. Gambar-gambar ini telah diproses menggunakan algoritma *face detection* Viola-Jones untuk memastikan wajah berada di posisi tengah dan dapat dikenali dengan konsisten. Dari seluruh individu yang ada, terdapat 1.683 orang yang memiliki dua atau lebih foto yang berbeda, yang memungkinkan untuk melakukan pengujian terhadap kemampuan sistem dalam mengenali wajah yang sama dengan variasi gambar. Dataset ini dirancang khusus untuk mendukung penelitian dalam bidang pengenalan wajah dalam kondisi yang bervariasi, seperti variasi pencahayaan, ekspresi wajah, sudut pandang, dan latar belakang yang berbeda-beda, sehingga dataset ini menjadi acuan standar dalam evaluasi model pengenalan wajah karena keragaman dan kualitas datanya yang representatif [19].

B Hasil Pengujian

Pengujian dilakukan untuk mengevaluasi performa sistem face recognition dalam mengenali dan memverifikasi wajah berdasarkan data yang telah ada. *Dataset Labeled Faces in the Wild* (LFW) digunakan sebagai bahan uji, mengingat kompleksitas dan keragaman gambar yang tersedia sangat ideal untuk menguji keakuratan serta ketahanan sistem terhadap variasi kondisi nyata, seperti pencahayaan, ekspresi, dan sudut wajah. Hasil dari pengujian menunjukkan bahwa sistem memiliki performa yang sangat baik dalam mencocokkan wajah. Berdasarkan data pengujian, total gambar yang diproses sebanyak 1.683 gambar, dengan hasil sebagai berikut.



Gambar 3.8. Hasil Pengujian Face Recognition Menggunakan Model FaceNet

Dari hasil pada Gambar 3.8 tersebut, dapat disimpulkan bahwa sistem memiliki tingkat akurasi sebesar 99% dan waktu eksekusi yang singkat untuk tiap gambar yang diproses. Hal ini menunjukkan bahwa sistem *face recognition* yang dikembangkan layak digunakan untuk implementasi verifikasi absensi karena mampu mengenali wajah dengan presisi tinggi dalam waktu yang relatif singkat.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan magang, terdapat beberapa kendala dan kesulitan yang dihadapi dalam proses pengembangan sistem *face recognition*. Adapun beberapa kendala yang ditemui adalah sebagai berikut.

1. Akurasi sistem *face recognition* belum optimal pada tahap awal karena gambar wajah yang digunakan masih merupakan gambar penuh tanpa proses pemotongan *(crop)*. Hal ini menyebabkan banyak noise pada input gambar yang mengurangi akurasi model *face recognition*.
2. Sistem pada awalnya tidak memiliki pengamanan untuk membatasi akses ke *endpoint* tertentu, sehingga terdapat potensi celah keamanan yang memungkinkan pengguna yang tidak terverifikasi mengakses fitur penting seperti *register* wajah baru. Hal ini menimbulkan kekhawatiran terhadap aspek otorisasi pengguna.

Untuk mengatasi kendala dan kesulitan tersebut, solusi yang diterapkan guna mendukung kelancaran pengembangan proyek adalah sebagai berikut.

1. Untuk meningkatkan akurasi sistem deteksi wajah, dilakukan *preprocessing* berupa *crop* otomatis pada wajah pengguna sebelum diproses lebih lanjut oleh model. Dengan melakukan *crop*, input yang diterima menjadi lebih bersih dan fokus pada area wajah, sehingga akurasi model meningkat secara signifikan.

2. Dalam upaya memperkuat aspek keamanan sistem, diterapkan mekanisme autentikasi dan otorisasi menggunakan *JSON Web Token* (JWT). Dengan penggunaan JWT, hanya pengguna yang telah berhasil login dan memiliki token valid yang dapat mengakses *endpoint* tertentu. Selain itu, role pengguna seperti admin juga disimpan di dalam token untuk membatasi akses ke fitur-fitur tertentu secara lebih ketat.