

MAKALAH
ARSITEKTUR PERANGKAT LUNAK
DESAIN ARSITEKTUR SISTEM PRESENSI MAHASISWA
BERBASIS FACE RECOGNITION
DENGAN PENDEKATAN EVENT-DRIVEN ARCHITECTURE (EDA)

Dosen Pengampuh : Mardiyyah Hasnawi. S.Kom., M.T., MTA.



Disusun Oleh :

Hendrawan 13020230309

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS MUSLIM INDONESIA
MAKASSAR
2025

DAFTAR ISI

BAB I	4
PENDAHULUAN	4
1.1 Latar Belakang	4
1.2 Kebutuhan Bisnis.....	4
1.3 Stakeholders	5
1.4 Scope.....	5
BAB II.....	7
ARCHITECTURE THINKING	7
2.1 Business Drivers	7
2.2 Trade-offs Analysis.....	7
2.3 Tanggung Jawab	8
2.4 Balance	9
BAB III	10
KARAKTERISTIK ARSITEKTUR	10
3.1 Penjelasan Umum	10
3.2 Karakteristik Arsitektur	10
3.3 Penjelasan dan Keterkaitan dengan Business Drivers.....	11
3.4 Analisis Hubungan Antar Karakteristik	11
BAB IV	13
GAYA ARSITEKTUR.....	13
4.1 Proses Pemilihan Gaya Arsitektur	13
4.2 Diagram Arsitektur Sistem	14
4.3 Best Practices	15
4.4 Trade-offs	15
BAB V	17
PRINSIP ARSITEKTUR.....	17
5.1 Modularitas.....	17
5.2 Separation of Concerns	18
5.3 SOLID	18
BAB VI.....	20
POLA ARSITEKTUR & DESIGN PATTERN	20
6.1 Pola Arsitektur	20
6.1.1 Event Bus Pattern	20
6.2 Design Pattern.....	20
6.2.1 Factory Pattern	20
6.2.2 Observer Pattern.....	20
6.2.3 Strategy Pattern.....	21
6.3 UML Lengkap untuk Satu Pattern — Observer.....	21
6.3.1 PlantUML — Class Diagram (Observer)	21
6.3.2 PlantUML — Sequence Diagram (Notifikasi setelah data ingest)	21

6.4 Penutup Bagian Pola & Pattern	22
BAB VII.....	23
PENUTUP	23
7.1 Kesimpulan.....	23
7.2 Rekomendasi Implementasi	24
7.3 Risiko dan Mitigasi	24
7.4 Metrik Keberhasilan Sistem.....	24
7.5 Penutup	25
DAFTAR PUSTAKA	25

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat mendorong perguruan tinggi untuk melakukan digitalisasi di berbagai aspek kegiatan akademik. Salah satu aspek penting yang memerlukan inovasi adalah sistem presensi mahasiswa. Selama ini, proses kehadiran di sebagian besar fakultas di Universitas Muslim Indonesia (UMI) masih dilakukan secara manual melalui tanda tangan. Kondisi ini menimbulkan permasalahan dalam hal akurasi data kehadiran, efisiensi waktu perkuliahan, serta kesulitan dalam pelaporan kehadiran untuk kebutuhan akreditasi.

Untuk menjawab tantangan tersebut, diperlukan sebuah sistem presensi yang mampu mendeteksi wajah mahasiswa secara otomatis dan real-time, serta langsung menyimpan hasil verifikasi ke dalam basis data akademik. Teknologi Face Recognition (pengenalan wajah) dipilih karena memiliki tingkat keakuratan tinggi dan dapat memastikan bahwa mahasiswa yang terdeteksi benar-benar hadir secara fisik di kelas [1]. Agar sistem ini dapat berjalan secara efisien dan dapat menangani banyak proses secara bersamaan, digunakan pendekatan Event-Driven Architecture (EDA) [2]. Pendekatan ini memungkinkan sistem memproses setiap aktivitas sebagai event terpisah — mulai dari deteksi wajah, verifikasi identitas, hingga pencatatan kehadiran — tanpa saling menunggu proses lain [3]. Dengan demikian, sistem dapat bekerja secara asinkron, real-time, dan scalable, sesuai kebutuhan kampus menuju Smart Campus Digital UMI.

1.2 Kebutuhan Bisnis

Adapun kebutuhan utama yang diharapkan dari sistem ini adalah sebagai berikut:

- a. Validasi Kehadiran Otentik: Sistem harus mampu memastikan bahwa mahasiswa yang hadir adalah individu yang terdaftar secara sah melalui pengenalan wajah otomatis .
- b. Integrasi Otomatis dengan Sistem Akademik (SIAKAD): Data kehadiran mahasiswa dapat langsung tersimpan ke sistem akademik tanpa proses input manual.
- c. Pelaporan Real-Time: Dosen dan admin fakultas dapat melihat status kehadiran mahasiswa secara langsung melalui dashboard interaktif [4].

Adapun kebutuhan fungsional dari sistem yang diusulkan adalah sebagai berikut:

1. Sistem dapat mendeteksi wajah mahasiswa melalui kamera dan mengirim hasil deteksi ke server pusat.
2. Sistem menampilkan dashboard kehadiran bagi dosen dan admin fakultas.
3. Sistem menyimpan log aktivitas presensi untuk pelacakan dan audit.

Adapun kebutuhan Non-fungsional dari sistem yang diusulkan adalah sebagai berikut:

1. Keamanan (Security): Data wajah mahasiswa harus terenkripsi dengan baik dan hanya dapat diakses oleh pihak berwenang.
2. Ketersediaan (Availability): Sistem harus tetap aktif selama jam perkuliahan dengan uptime minimal 99%.
3. Kinerja (Performance): Waktu proses deteksi hingga pencatatan kehadiran maksimal 2 detik.
4. Skalabilitas (Scalability): Sistem harus mampu menangani peningkatan jumlah mahasiswa setiap semester.
5. Kemudahan Penggunaan (Usability): Antarmuka sistem harus mudah digunakan oleh semua pihak, termasuk dosen dan mahasiswa.

1.3 Stakeholders

Stakeholders yang terlibat dalam sistem ini meliputi:

Stakeholder	Peran dan Kepentingan
Mahasiswa	Melakukan presensi otomatis melalui kamera di ruang kelas.
Dosen Pengampu	Memantau kehadiran mahasiswa secara real-time dan menghasilkan laporan kehadiran.
Admin Fakultas / Program Studi	Mengelola data kehadiran, validasi hasil deteksi, dan rekap kehadiran perkuliahan.
Bagian IT / Pusat Data Kampus	Menyediakan infrastruktur server, memastikan keamanan data, serta melakukan pemeliharaan sistem.
Wakil Dekan III / Biro Akademik	Menggunakan data kehadiran sebagai bahan evaluasi dan laporan akreditasi fakultas.

1.4 Scope

Ruang lingkup pengembangan sistem mencakup:

1. Pendekripsi dan pengenalan wajah mahasiswa menggunakan algoritma Face Recognition.
2. Verifikasi identitas mahasiswa dan pencatatan otomatis ke dalam database presensi.
3. Integrasi hasil presensi dengan sistem akademik (SIAKAD) kampus.
4. Penyediaan dashboard monitoring untuk dosen dan admin fakultas.

Batasan sistem yang dikembangkan adalah sebagai berikut:

1. Sistem tidak mencakup manajemen nilai akademik atau materi pembelajaran (LMS).
2. Sistem difokuskan pada presensi mahasiswa, bukan presensi dosen atau staf.
3. Proses training model wajah dilakukan sebelumnya, bukan secara langsung saat presensi.
4. Data biometrik mahasiswa disimpan secara terenkripsi untuk menjaga privasi.

BAB II

ARCHITECTURE THINKING

2.1 Business Drivers

Pengembangan sistem presensi berbasis Face Recognition di Universitas Muslim Indonesia (UMI) didasari oleh kebutuhan kampus untuk meningkatkan efisiensi proses akademik, validasi kehadiran yang akurat, dan integrasi data presensi ke sistem akademik (SIAKAD). Sistem ini dirancang untuk menjawab tantangan pengelolaan data kehadiran yang masih manual dan rawan kecurangan, serta mendukung transformasi menuju Smart Campus Digital [5].

Tabel berikut menjelaskan tiga business driver utama yang berpengaruh langsung terhadap keputusan arsitektur sistem:

No	Business Driver	Penjelasan	Dampak terhadap Arsitektur
1	Efisiensi Proses Presensi	Kampus membutuhkan sistem otomatis yang mampu mencatat kehadiran tanpa intervensi manual dosen.	Membutuhkan arsitektur yang mendukung pemrosesan event secara asinkron dan cepat.
2	Keamanan Data Biometrik	Data wajah mahasiswa bersifat sensitif dan harus dijaga kerahasiaannya.	Arsitektur harus mendukung enkripsi data, autentifikasi pengguna, dan pembatasan akses.
3	Integrasi Akademik dan Skalabilitas Sistem	Sistem presensi harus dapat berkomunikasi dengan SIAKAD dan mampu menangani peningkatan jumlah pengguna.	Membutuhkan API Gateway dan event broker untuk mendukung integrasi modular dan horizontal scaling.

Business drivers tersebut menjadi dasar pemilihan Event-Driven Architecture (EDA) sebagai pendekatan utama, karena memungkinkan sistem bekerja secara modular, real-time, dan terdistribusi, sesuai kebutuhan UMI yang terus berkembang [2].

2.2 Trade-offs Analysis

Dalam proses perancangan arsitektur, tim pengembang melakukan analisis **trade-offs** antara beberapa alternatif solusi untuk mencapai keseimbangan antara kinerja, biaya, dan kompleksitas sistem.

Aspek	Opsi A	Opsi B	Keputusan	Alasan Pemilihan
-------	--------	--------	-----------	------------------

Metode Presensi	QR Code	Face Recognition	<input checked="" type="checkbox"/> Face Recognition	Lebih akurat, tidak bisa ditip, dan mendukung identifikasi otomatis.
Gaya Arsitektur	Monolithic Architecture	Event-Driven Architecture	<input checked="" type="checkbox"/> Event-Driven	Lebih fleksibel, scalable, dan cocok untuk proses real-time event.
Infrastruktur	Server Lokal	Cloud-Based Infrastructure	<input checked="" type="checkbox"/> Cloud-Based	Mendukung ketersediaan tinggi (availability) dan akses lintas fakultas.
Integrasi Sistem	Manual Input	API Gateway & Message Queue	<input checked="" type="checkbox"/> API Gateway	Sinkronisasi data otomatis ke SIAKAD, efisien, dan mudah dikontrol.
Komunikasi Antar Modul	Sinkron	Asinkron (Event Broker)	<input checked="" type="checkbox"/> Asinkron	Tidak saling menunggu proses, meningkatkan throughput sistem.

Dari hasil analisis di atas, tim memutuskan bahwa pendekatan Face Recognition + Event-Driven Architecture + Cloud Infrastructure adalah kombinasi terbaik untuk memenuhi kebutuhan bisnis dan non-fungsional kampus.

2.3 Tanggung Jawab

Proyek ini dikerjakan oleh dua peran utama: Arsitek Sistem dan Developer. Pembagian tanggung jawab dilakukan untuk menjaga kejelasan peran dalam siklus pengembangan perangkat lunak.

Aktivitas / Keputusan	Architect	Developer
Desain struktur arsitektur sistem (EDA)	✓	✗
Pemilihan dan konfigurasi event broker	✓	✓
Implementasi algoritma face recognition	✗	✓
Integrasi sistem dengan SIAKAD	✓	✓

Pengujian dan validasi performa sistem	X	✓
Deployment dan monitoring cloud server	✓	✓
Dokumentasi dan laporan arsitektur	✓	X

Dengan pembagian tanggung jawab ini, proses pengembangan menjadi lebih efisien karena setiap pihak fokus pada kompetensinya masing-masing. Arsitek sistem berperan dalam perancangan dan pengambilan keputusan strategis, sementara developer berfokus pada implementasi teknis dan integrasi sistem.

2.4 Balance

Dalam perancangan sistem berbasis Event-Driven Architecture, beberapa faktor keseimbangan harus diperhatikan agar sistem tetap efisien dan mudah dikelola dalam jangka panjang. Berikut adalah beberapa trade-off balance yang dipertimbangkan:

1. Performa vs Biaya

Sistem berbasis cloud memiliki performa tinggi dan akses cepat, namun menambah biaya infrastruktur. Untuk menyeimbangkan hal ini, digunakan layanan cloud hybrid dengan auto-scaling agar sumber daya aktif hanya saat dibutuhkan.

2. Kompleksitas vs Maintainability

EDA lebih kompleks dibandingkan arsitektur monolitik karena banyak event handler dan service terpisah. Namun, modularitas sistem ini meningkatkan kemudahan perawatan (maintainability) dan pengembangan fitur di masa depan.

3. Time vs Quality

Pengembangan model AI untuk face recognition memerlukan waktu pelatihan (training) yang cukup lama. Untuk menjaga kualitas tanpa menghambat jadwal proyek, proses training dilakukan paralel dengan pengembangan sistem antarmuka.

4. Security vs Usability

Tingkat keamanan tinggi melalui autentikasi dan enkripsi dapat menambah langkah akses pengguna. Solusinya, sistem menerapkan Single Sign-On (SSO) untuk menjaga keamanan tanpa mengurangi kenyamanan pengguna.

BAB III

KARAKTERISTIK ARSITEKTUR

3.1 Penjelasan Umum

Setiap sistem perangkat lunak memiliki karakteristik arsitektur yang menentukan kualitas, keandalan, serta kemampuannya untuk berkembang. Dalam konteks Sistem Presensi Mahasiswa Berbasis Face Recognition di Universitas Muslim Indonesia (UMI), arsitektur harus mampu memenuhi kebutuhan keamanan data biometrik, kecepatan deteksi wajah, serta integrasi dengan sistem akademik kampus.

Untuk itu, tim pengembang menetapkan lima karakteristik utama yang menjadi prioritas dalam perancangan sistem. Pemilihan ini didasarkan pada *business drivers* yang telah dijelaskan sebelumnya, serta kebutuhan operasional kampus terhadap sistem presensi yang efisien, aman, dan real-time.

3.2 Karakteristik Arsitektur

Karakteristik	Kategori	Prioritas	Justifikasi	Metrik Target
Availability	Operational	HIGH	Sistem harus selalu dapat digunakan selama jam perkuliahan tanpa downtime.	99.9% uptime selama semester berjalan.
Security	Cross-Cutting	HIGH	Data wajah mahasiswa bersifat sensitif, sehingga perlu perlindungan enkripsi dan autentikasi akses.	Implementasi AES-256 Encryption dan SSO login.
Performance	Operational	HIGH	Presensi wajah harus diproses dengan cepat agar tidak mengganggu proses belajar mengajar.	Waktu respon < 2 detik per deteksi wajah.
Scalability	Structural	MEDIUM	Sistem harus dapat menangani peningkatan jumlah mahasiswa dan fakultas tanpa penurunan performa.	Dapat menangani 1000+ request per menit.
Usability	Operational	MEDIUM	Sistem digunakan oleh berbagai pengguna	80% pengguna dapat memahami sistem

			(dosen, admin, mahasiswa), sehingga antarmuka harus intuitif.	tanpa pelatihan tambahan.
--	--	--	---	---------------------------

3.3 Penjelasan dan Keterkaitan dengan Business Drivers

1. Availability (Ketersediaan Sistem)

Karakteristik ini diprioritaskan tinggi karena sistem presensi digunakan secara aktif pada setiap sesi perkuliahan. Downtime sekecil apa pun dapat mengganggu proses akademik. Untuk menjamin ketersediaan, sistem diimplementasikan pada infrastruktur **cloud dengan auto-scaling dan load balancing**.

2. Security (Keamanan Data Biometrik)

Keamanan menjadi aspek kritis mengingat data wajah mahasiswa termasuk **data pribadi sensitif**. Sistem menerapkan **enkripsi AES-256** untuk penyimpanan data wajah dan autentikasi pengguna berbasis token untuk mencegah akses ilegal.

3. Performance (Kinerja Sistem)

Dalam sistem real-time, performa merupakan faktor utama. Arsitektur **Event-Driven** memungkinkan proses deteksi, verifikasi, dan pencatatan berlangsung paralel tanpa menunggu proses lain. Hal ini memastikan sistem tetap responsif meskipun banyak pengguna melakukan presensi secara bersamaan.

4. Scalability (Skalabilitas Sistem)

Karena jumlah mahasiswa di UMI terus meningkat setiap tahun, sistem perlu didesain agar dapat dikembangkan dengan mudah. Dengan pendekatan EDA dan penerapan **message broker (seperti Kafka atau RabbitMQ)**, sistem dapat menangani ribuan event presensi secara simultan.

5. Usability (Kemudahan Penggunaan)

Pengguna sistem berasal dari berbagai latar belakang — mahasiswa, dosen, dan admin fakultas — sehingga antarmuka harus mudah dipahami. Dashboard dirancang dengan tampilan sederhana dan berbasis web agar dapat diakses dari berbagai perangkat.

3.4 Analisis Hubungan Antar Karakteristik

Karakteristik-karakteristik tersebut saling berkaitan dan saling memengaruhi dalam implementasinya. Misalnya, peningkatan security dapat berdampak pada penurunan performance, karena proses enkripsi menambah beban komputasi. Untuk menyeimbangkannya, sistem menggunakan caching dan load balancing agar tetap efisien tanpa mengorbankan keamanan.

Begitu pula antara availability dan scalability, yang sama-sama dijaga melalui pemanfaatan arsitektur event-driven berbasis cloud, memungkinkan sistem beroperasi secara terus-menerus dan berkembang seiring pertumbuhan data.

BAB IV

GAYA ARSITEKTUR

4.1 Proses Pemilihan Gaya Arsitektur

Dalam perancangan sistem presensi berbasis Face Recognition, pemilihan gaya arsitektur tidak dapat dilakukan secara sembarangan. Tahap pemilihan dilakukan melalui proses eliminasi, evaluasi, dan keputusan akhir, seperti berikut:

1. Eliminasi Gaya yang Tidak Cocok

Gaya Arsitektur	Alasan Eliminasi
Layered Architecture	Kurang sesuai untuk sistem real-time yang memerlukan pemrosesan paralel dan cepat. Terlalu banyak lapisan membuat respon lebih lambat.
Monolithic Architecture	Tidak fleksibel untuk dikembangkan atau diskalakan saat jumlah event (presensi) meningkat.
SOA (Service-Oriented Architecture)	Cocok untuk integrasi antar layanan besar, namun overhead komunikasi SOAP/REST terlalu berat untuk event cepat seperti deteksi wajah.
Microservices Architecture	Meski fleksibel, terlalu kompleks untuk skala awal sistem presensi yang berfokus pada event streaming dan real-time response.

2. Evaluasi Gaya yang Tersisa

Dari hasil eliminasi, dua gaya yang masih relevan adalah Event-Driven Architecture (EDA) dan Hybrid (EDA + Microservices).

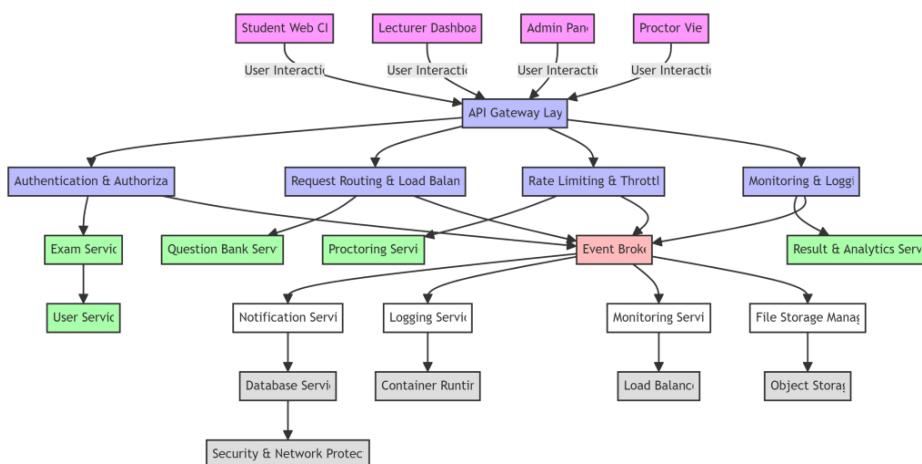
Kriteria	EDA	Hybrid EDA + Microservices
Real-time event handling	✓ Sangat efisien	✓ Efisien
Kompleksitas awal	✓ Rendah	✗ Lebih tinggi
Skalabilitas	✓ Tinggi	✓ Tinggi
Maintenance	✓ Mudah	✗ Lebih rumit
Biaya operasional	✓ Lebih rendah	✗ Lebih mahal

3. Keputusan Final

Setelah mempertimbangkan kebutuhan performa real-time, fleksibilitas, dan efisiensi biaya, maka dipilih gaya Event-Driven Architecture (EDA) sebagai arsitektur utama sistem ini.

EDA memungkinkan sistem untuk merespons setiap event (misalnya "wajah terdeteksi", "verifikasi berhasil", "kehadiran dicatat") secara asinkron dan terdistribusi, tanpa harus menunggu proses lain selesai.

4.2 Diagram Arsitektur Sistem



Penjelasan Komponen :

1. Client Layer

- Student Web CBT: Platform untuk mahasiswa untuk login dan mengikuti ujian.
- Lecturer Dashboard: Dasbor dosen untuk memantau kehadiran dan mengelola ujian.
- Admin Panel: Panel admin untuk mengelola data pengguna dan konfigurasi.
- Proctor View: Tampilan pengawas untuk memonitor ujian mahasiswa.

2. API Gateway Layer

- Authentication & Authorization: Verifikasi identitas dan kontrol akses pengguna.
- Request Routing & Load Balancing: Menyaring dan mendistribusikan permintaan pengguna ke layanan yang sesuai.
- Rate Limiting & Throttling: Membatasi jumlah permintaan untuk mencegah penyalahgunaan.
- Monitoring & Logging: Pemantauan kinerja sistem dan pencatatan log untuk analisis.

3. Microservices Layer

- Exam Service: Mengelola soal ujian dan pengumpulan jawaban.
- Question Bank Service: Mengelola soal ujian dan kategorinya.
- Proctoring Service: Memantau ujian secara real-time.
- Result & Analytics Service: Mengelola hasil ujian dan analitik.

- User Service: Menyimpan data pengguna (mahasiswa, dosen, admin).

4. Message Broker Layer

- Event Broker (Kafka/RabbitMQ): Mengirim dan menerima event antar layanan secara asinkron.

5. Support Services Layer

- Notification Service: Mengirim notifikasi (email, SMS).
- Logging Service: Mencatat log sistem untuk audit dan pemeliharaan.
- Monitoring Service: Memantau status dan kinerja sistem.
- File Storage Manager: Mengelola penyimpanan file soal ujian dan jawaban.

6. Infrastructure Layer

- Database Service: Menyimpan data seperti mahasiswa, ujian, dan hasil.
- Container Runtime: Menjalankan aplikasi di dalam container (misal, Docker).
- Load Balancer: Menyebarluaskan beban server agar tetap responsif.
- Object Storage: Penyimpanan file besar (misal soal ujian, video).
- Security & Network Protection: Mengamankan sistem dan data dari ancaman eksternal.

4.3 Best Practices

Best Practice	Penjelasan
1. Gunakan Message Broker untuk Asynchronous Processing	Menghindari blocking antar service dengan memanfaatkan event queue seperti Kafka atau RabbitMQ.
2. Idempotency pada Consumer Service	Setiap event yang diterima harus memiliki ID unik agar tidak terjadi duplikasi pencatatan kehadiran.
3. Event Logging & Monitoring	Setiap event dicatat (logging) agar mudah dilakukan audit dan debugging jika terjadi anomali.
4. Loose Coupling antar Komponen	Komponen Face Recognition, Attendance, dan Dashboard tidak saling tergantung langsung, hanya berkomunikasi melalui event.
5. Fault Tolerance Design	Sistem harus mampu menampung event sementara saat terjadi gangguan jaringan, lalu memproses ulang ketika sistem normal.

4.4 Trade-offs

Aspek	Keuntungan (EDA)	Kelemahan / Risiko	Mitigasi

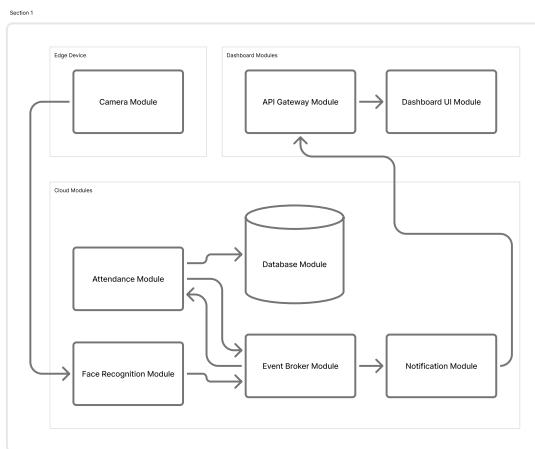
Performa	Pemrosesan cepat dan real-time karena asinkron.	Kompleksitas debugging event yang sulit dilacak.	Implementasi logging & event tracing (Elastic Stack).
Kompleksitas Sistem	Skalabilitas tinggi dan mudah dikembangkan.	Membutuhkan message broker tambahan (Kafka/RabbitMQ).	Gunakan cloud-managed broker untuk mengurangi beban konfigurasi.
Konsistensi Data	High availability, tidak perlu menunggu transaksi lain.	Potensi <i>event loss</i> atau <i>event duplication</i> .	Gunakan pola At-Least-Once Delivery dan mekanisme kompensasi data.

BAB V

PRINSIP ARSITEKTUR

5.1 Modularitas

Diagram modular :



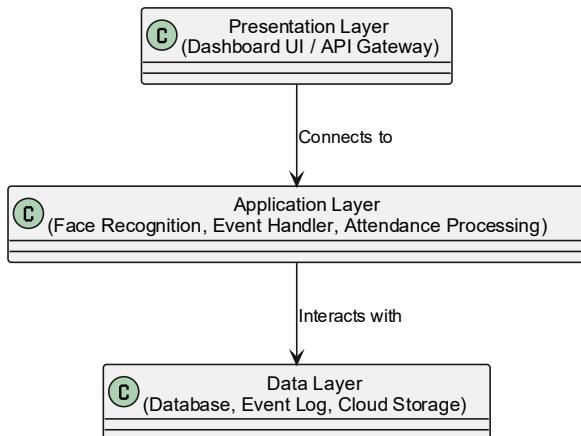
Modularitas adalah prinsip dasar arsitektur perangkat lunak yang bertujuan untuk membagi sistem menjadi komponen-komponen independen agar mudah dikembangkan, diuji, dan dipelihara.

Dalam sistem presensi berbasis Face Recognition ini, modularitas diterapkan dengan memisahkan setiap fungsi utama menjadi modul tersendiri yang saling berinteraksi melalui event broker (message queue).

Pembagian Modul Sistem:

Modul	Fungsi Utama	Keterangan
Camera Module	Menangkap wajah mahasiswa dan mengirim gambar ke sistem.	Dijalankan di perangkat edge (kelas).
Face Recognition Module	Melakukan deteksi dan verifikasi wajah.	Menggunakan model AI (CNN / dlib).
Event Broker Module	Menyalurkan event antar modul secara asinkron.	Menggunakan RabbitMQ / Kafka.
Attendance Module	Menyimpan hasil presensi ke database dan menghasilkan event baru.	Berfungsi sebagai consumer event.
Dashboard Module	Menampilkan laporan kehadiran real-time.	Mengakses API Gateway.

5.2 Separation of Concerns



Penjelasan Layer:

1. Presentation Layer

- Bertanggung jawab menampilkan informasi kehadiran mahasiswa dan status sistem secara real-time.
- Dapat diakses oleh dosen dan admin melalui antarmuka web.

2. Application Layer

- Menangani seluruh logika bisnis utama: deteksi wajah, verifikasi identitas, pengiriman event, dan pencatatan kehadiran.
- Menggunakan arsitektur event-driven untuk komunikasi antar modul.

3. Data Layer

- Mengelola penyimpanan data wajah, hasil presensi, serta event log.
- Menerapkan enkripsi pada data biometrik untuk menjaga keamanan.

5.3 SOLID

Prinsip 1: Single Responsibility Principle (SRP)

Contoh Kasus:

Kelas FaceDetector hanya bertanggung jawab untuk mendeteksi wajah, bukan untuk menyimpan data kehadiran atau mengirim notifikasi.

Implementasi:

```
class FaceDetector:
```

```
    def detect_face(self, frame):  
        # memproses citra untuk mendeteksi wajah
```

```
return detected_faces
```

Benefit:

- Kode lebih mudah diuji dan dipelihara.
- Meminimalkan bug akibat tanggung jawab ganda dalam satu kelas.

Prinsip 2: Open-Closed Principle (OCP)

Contoh Kasus:

Sistem harus dapat menambahkan algoritma pengenalan wajah baru tanpa mengubah struktur inti.

Implementasi:

```
class RecognitionStrategy:
```

```
    def recognize(self, image): pass
```

```
class CNNRecognition(RecognitionStrategy):
```

```
    def recognize(self, image):
```

```
        # implementasi dengan CNN
```

```
        pass
```

```
class HOGRecognition(RecognitionStrategy):
```

```
    def recognize(self, image):
```

```
        # implementasi dengan HOG
```

```
        Pass
```

Benefit:

- Sistem dapat diperluas tanpa memodifikasi kode utama.
- Memudahkan pembaruan algoritma AI di masa depan.

BAB VI

POLA ARSITEKTUR & DESIGN PATTERN

6.1 Pola Arsitektur

6.1.1 Event Bus Pattern

Aspek	Deskripsi
Masalah	Komponen sistem (camera, recognition, attendance, dashboard) perlu berkomunikasi tanpa saling ketergantungan langsung. Jika komunikasi dilakukan secara sinkron (HTTP), maka sistem menjadi lambat dan tidak scalable.
Solusi	Gunakan <i>Event Bus (Message Broker)</i> sebagai perantara. Setiap layanan mengirim event dan layanan lain dapat “subscribe” tanpa harus tahu sumbernya.
Implementasi	Menggunakan RabbitMQ atau Apache Kafka sebagai <i>event broker</i> . Saat wajah mahasiswa terdeteksi, FaceRecognitionService menerbitkan event FaceVerified, dan AttendanceService otomatis memprosesnya.
Benefit	- Loose coupling antar modul.- Mendukung <i>asynchronous processing</i> .- Meningkatkan performa sistem real-time.

6.2 Design Pattern

Dalam pengembangan berbasis OOP, sistem ini menerapkan tiga design pattern utama yang mendukung arsitektur event-driven agar tetap efisien dan mudah dikembangkan [6].

6.2.1 Factory Pattern

Aspek	Deskripsi
Masalah	Sistem menggunakan lebih dari satu metode deteksi wajah (misal CNN dan HOG). Dibutuhkan cara fleksibel untuk membuat objek algoritma sesuai konfigurasi.
Solusi	Gunakan Factory Pattern untuk membuat objek <i>FaceRecognitionAlgorithm</i> tanpa harus menentukan kelas spesifik di kode utama.

6.2.2 Observer Pattern

Aspek	Deskripsi

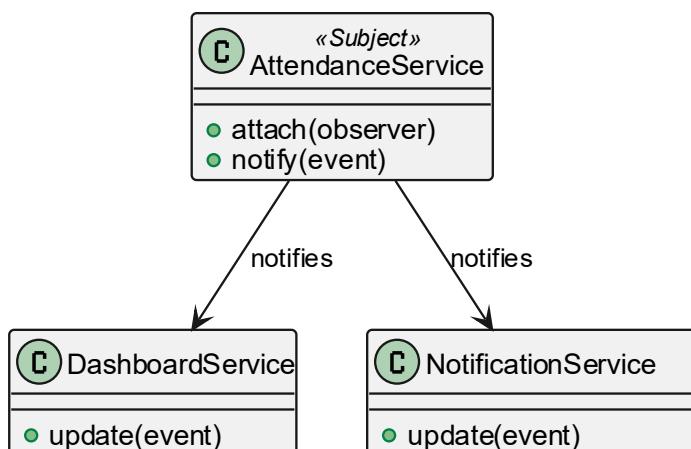
Masalah	Sistem membutuhkan mekanisme notifikasi otomatis setelah presensi berhasil dicatat agar dashboard dan admin segera menerima pembaruan.
Solusi	Terapkan Observer Pattern, di mana <i>AttendanceService</i> bertindak sebagai Subject, dan <i>DashboardService</i> serta <i>NotificationService</i> sebagai Observers.

6.2.3 Strategy Pattern

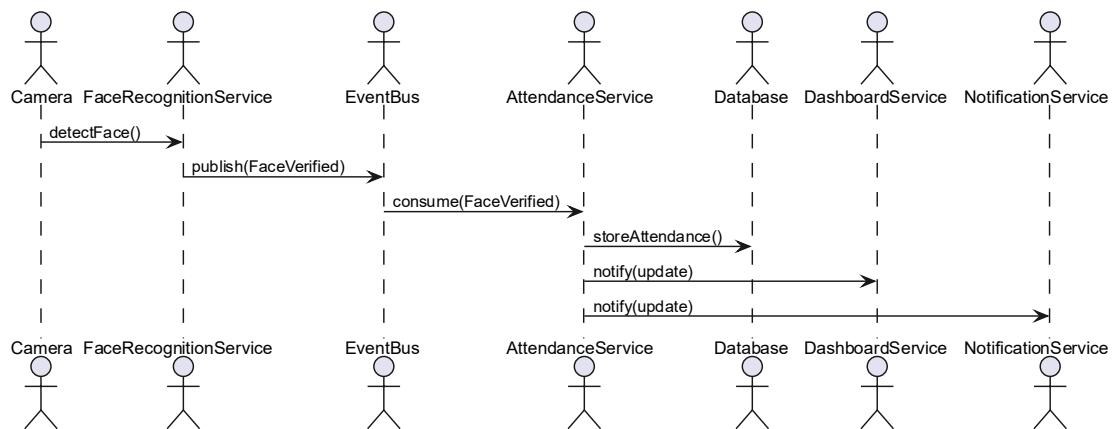
Aspek	Deskripsi
Masalah	Sistem perlu mendukung berbagai strategi autentikasi (misalnya Face Recognition, QR Code, atau PIN darurat).
Solusi	Gunakan Strategy Pattern agar sistem dapat berganti metode autentikasi tanpa mengubah logika utama.

6.3 UML Lengkap untuk Satu Pattern — Observer

6.3.1 PlantUML — Class Diagram (Observer)



6.3.2 PlantUML — Sequence Diagram (Notifikasi setelah data ingest)



6.4 Penutup Bagian Pola & Pattern

Penggunaan kombinasi Event Bus, API Gateway, dan tiga Design Pattern (Factory, Observer, Strategy) menjadikan sistem ini fleksibel, modular, dan mudah beradaptasi dengan perubahan kebutuhan.

Pola-pola ini juga mendukung prinsip arsitektur Event-Driven yang memisahkan komponen secara longgar, memungkinkan sistem bekerja asinkron, resilien, dan mudah diskalakan di masa depan.

BAB VII

PENUTUP

7.1 Kesimpulan

Perancangan Sistem Presensi Mahasiswa Berbasis Face Recognition dengan Pendekatan Event-Driven Architecture (EDA) bertujuan untuk meningkatkan efisiensi, keakuratan, dan keandalan proses kehadiran mahasiswa di lingkungan Universitas Muslim Indonesia (UMI). Sistem ini dibangun untuk menjawab berbagai permasalahan pada metode presensi konvensional seperti *kecurangan presensi (titip absen)*, keterlambatan rekap data, serta ketidakefisienan integrasi antar sistem akademik.

Dengan menerapkan Event-Driven Architecture, setiap proses utama — mulai dari deteksi wajah, verifikasi identitas, pencatatan presensi, hingga pembaruan dashboard — berjalan secara asinkron dan independen, tanpa menunggu proses lain selesai. Hal ini membuat sistem mampu menangani banyak pengguna secara bersamaan dengan latensi yang sangat rendah.

Beberapa keputusan arsitektur penting yang diambil dalam sistem ini antara lain:

- Penggunaan Event Bus Pattern (RabbitMQ / Kafka) untuk komunikasi antar modul secara asinkron.
- Implementasi API Gateway Pattern untuk mengonsolidasikan akses data melalui satu pintu.
- Penerapan Design Pattern seperti Factory, Observer, dan Strategy untuk menjamin fleksibilitas dan keterpisahan tanggung jawab dalam sistem.
- Penerapan karakteristik arsitektur utama seperti *availability, security, performance, scalability, dan usability* untuk memastikan sistem tetap stabil, aman, dan mudah digunakan.

Dengan pendekatan ini, sistem presensi mampu:

1. Memastikan kehadiran mahasiswa secara autentik dan real-time.
2. Menghasilkan laporan presensi otomatis yang siap digunakan untuk keperluan akademik dan akreditasi.
3. Mendukung pengembangan jangka panjang menuju Smart Campus Digital di UMI.

7.2 Rekomendasi Implementasi

Fase	Rekomendasi Implementasi
Fase 1 – Prototyping	Mengembangkan versi awal face recognition menggunakan dataset mahasiswa terbatas dan menguji akurasi model AI.
Fase 2 – Integrasi Sistem	Menghubungkan Face Recognition Service dengan Event Broker (RabbitMQ / Kafka) dan Attendance Database.
Fase 3 – Deployment	Menyebarluaskan sistem pada server kampus dengan dukungan API Gateway dan dashboard admin.
Fase 4 – Evaluasi & Optimalisasi	Melakukan uji performa, keandalan sistem, dan keamanan data biometrik.
Fase 5 – Ekspansi	Mengintegrasikan modul ini dengan sistem lain seperti e-learning, SIM akademik, dan kepegawaian.

7.3 Risiko dan Mitigasi

Risiko Potensial	Dampak	Strategi Mitigasi
Gangguan jaringan saat proses presensi.	Kegagalan event terkirim atau data tidak tersimpan.	Menyimpan event sementara (message queue retry) dan proses ulang otomatis.
Kesalahan deteksi wajah (false positive/negative).	Presensi salah atau tidak terbaca.	Menambah dataset pelatihan model AI dan menggunakan teknik ensemble model.
Ancaman keamanan data biometrik.	Kebocoran data wajah mahasiswa.	Menggunakan enkripsi AES-256, autentikasi token JWT, dan firewall API Gateway.
Kinerja server berkurang saat beban tinggi.	Respon sistem melambat.	Menerapkan autoscaling dan load balancing di cloud.

7.4 Metrik Keberhasilan Sistem

Aspek	Indikator Keberhasilan	Target
Kinerja (Performance)	Waktu deteksi dan pencatatan presensi	≤ 2 detik per event
Keandalan (Availability)	Sistem tersedia selama jam kuliah	99.9% uptime

Akurasi Accuracy)	(Recognition	Ketepatan deteksi wajah mahasiswa	$\geq 95\%$
Keamanan (Security)		Tidak ada kebocoran data selama uji coba	100% aman
Kepuasan Pengguna (Usability)		Nilai kepuasan dosen dan admin	$\geq 80\%$ positif

7.5 Penutup

Dengan desain arsitektur yang kuat, modular, dan berbasis event-driven, sistem Smart Attendance Face Recognition di UMI tidak hanya memecahkan masalah presensi manual, tetapi juga membuka peluang besar menuju digitalisasi kampus yang lebih cerdas, efisien, dan terintegrasi [7].

Sistem ini dapat menjadi *fondasi awal* bagi pengembangan ekosistem Smart Campus Digital UMI, sekaligus mendukung visi universitas dalam membangun pendidikan berbasis teknologi informasi dan inovasi berkelanjutan.

DAFTAR PUSTAKA

- [1] M. A. Khair, P. Aldiyuda, N. Enjelina P, M. Z. Zukhrufa, and M. Adrezo, "Perancangan Sistem Absensi Mahasiswa Berbasis Face Recognition di Lingkungan UPN Veteran Jakarta," *Inform. J. Ilmu Komput.*, vol. 20, no. 1, pp. 35–42, 2024, doi: 10.52958/iftk.v20i1.6696.
- [2] R. Kumar, "Event-Driven Architectures for Real-Time Data Processing: A Deep Dive into System Design and Optimization," *evolution*. researchgate.net.
- [3] K. Alhanaee, M. Alhammadi, N. Almenhali, and M. Shatnawi, "Face Recognition Smart Attendance System using Deep Transfer Learning," *Procedia Comput. Sci.*, vol. 192, pp. 4093–4102, 2021, doi: <https://doi.org/10.1016/j.procs.2021.09.184>.
- [4] N. Dumaturun and R. M. N. Betaubun, "Sistem Absensi Berbasis Digital Dalam Meningkatkan Disiplin Pegawai Pada Dinas Pendapatan Daerah Bagian Sekretariat Kabupaten Merauke," *Papsel Economic Journal*. 2025.
- [5] H. Al Akbar, M. R. Faturrahman, and S. Sidharta, "Guidance in Designing A Smart Campus: A Systematic Literature Review," *Procedia Comput. Sci.*, vol. 227, pp. 83–91, 2023, doi: <https://doi.org/10.1016/j.procs.2023.10.505>.
- [6] Gregor Hohpe and Martin Fowler, *Agile EAI - Enterprise Integration Patterns*. 2003. [Online]. Available: <https://www.enterpriseintegrationpatterns.com/docs/agileEAI.html>
- [7] C. Fernando, *Solution architecture patterns for enterprise: A guide to building enterprise software systems*. 2022. doi: 10.1007/978-1-4842-8948-8.