BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Konsep Dasar *Website* dan Sistem Pelacakan Keberadaan Dosen

Perkembangan teknologi informasi saat ini telah mendorong transformasi digital di berbagai sektor, termasuk di lingkungan institusi pendidikan tinggi. Sistem informasi konvensional secara bertahap mulai digantikan oleh infrastruktur perangkat lunak modern yang mampu memfasilitasi pertukaran informasi secara instan, aman, dan terpusat. Menurut (Putriyani, 2025:325), penggunaan *framework web* dapat meningkatkan integrasi antarmodul dan efisiensi pengelolaan data dinamis dalam aplikasi berbasis *web*. Untuk mewujudkan ekosistem digital yang efisien, diperlukan landasan platform yang tangguh serta mekanisme pemutakhiran data yang memiliki tingkat keandalan tinggi. Penggabungan kedua elemen konseptual ini tidak hanya berfungsi untuk mempercepat distribusi informasi akademik, tetapi juga memberikan jaminan transparansi data operasional harian. Oleh karena itu, penggunaan sistem berbasis *web* dapat meningkatkan efisiensi proses, menyajikan informasi kehadiran, serta mendukung digitalisasi administrasi akademik secara berkelanjutan Putriyani (2025:325).

a. *Website*

Website merupakan kumpulan halaman *web* yang saling terhubung dan dapat diakses melalui internet menggunakan peramban. Dalam literatur, *web*site dipandang sebagai pusat informasi digital yang dinamis dan interaktif, tanpa terikat ruang maupun waktu. Kehadirannya mengubah pola komunikasi, transaksi, serta pengelolaan operasional institusi dari pencatatan manual menuju sistem elektronik yang terpadu. Laudon dan Traver dalam (Malik, 2025:13) menjelaskan bahwa *website* merupakan platform digital yang berfungsi sebagai media informasi, komunikasi, serta transaksi daring bagi penggunanya.. Dalam konteks perangkat lunak Dosen *Tracker*, *web*site menjadi landasan utama interaksi operasional, khususnya untuk menampilkan status dan lokasi terakhir dosen secara terpusat.

Dengan demikian, *website* berperan sebagai fondasi integratif yang memungkinkan penyajian informasi ketersediaan dosen secara real-time, terpusat, dan mudah diakses oleh pengguna

b. Sistem Pelacakan Keberadaan Dosen

Sistem pelacakan keberadaan dosen berfungsi mengumpulkan dan menyebarkan informasi lokasi sivitas akademik agar mahasiswa dapat mengetahui posisi terakhir dosen di kampus. Berbeda dari sistem kehadiran administratif yang menekankan disiplin kepegawaian, Dosen *Tracker* berfokus pada penyediaan informasi lokasi secara real-time. Kajian sistem kehadiran digital menunjukkan bahwa pencatatan manual

rawan kesalahan atau manipulasi serta kurang efektif dan efisien (Lestari et al., 2024:338-339); (Meldyantono & Poetro, 2024:996); (Permatasari et al., 2024:114). Temuan itu memperkuat perlunya identifikasi otomatis, yang dalam penelitian ini diarahkan untuk memperbarui *last seen location* dosen dan menyampaikannya kepada mahasiswa.

Berdasarkan uraian tersebut, pelacakan keberadaan dosen diarahkan bukan sebagai alat pengawasan administratif, melainkan sebagai layanan informasi lokasi yang akurat, cepat, dan bermanfaat bagi mahasiswa.

2. Infrastruktur Komputasi Cerdas

Infrastruktur komputasi cerdas menjadi landasan arsitektur yang menyatukan sumber daya komputasi, penyimpanan, dan jaringan agar pemrosesan data pada era Internet of Things (IoT) berjalan efisien (Liu et al., 2019:1537). Infrastruktur ini dirancang untuk menangani volume data besar yang terus dihasilkan dan dianalisis oleh perangkat IoT (Laghari et al., 2022:2) . Tidak hanya bergantung pada satu pusat data, strukturnya bersifat terfederasi dan terdistribusi mulai dari perangkat terminal (*endpoint*), simpul tepi (*edge nodes*), hingga komputasi awan (Cao et al., 2020:85717). Komponen utama sistem IoT meliputi perangkat keras, perangkat lunak, jalur komunikasi, dan platform yang mendukung pengumpulan, pengiriman, serta pemrosesan data (Laghari et al., 2021:4)

a. *Edge Computing*

Edge computing merupakan paradigma komputasi yang memindahkan pemrosesan, penyimpanan, dan jaringan data ke tepi jaringan, lebih dekat ke perangkat pengguna atau sumber data (Liu et al., 2019:1537). Berbeda dari cloud computing yang memproses data secara terpusat dalam skala besar, edge computing menekankan pemrosesan lokal berskala lebih kecil (Cao et al., 2020:85716). Pendekatan ini meredam kelemahan cloud tradisional, seperti beban bandwidth tinggi, latensi lambat, dan konsumsi energi besar akibat pengiriman data terus-menerus (Cao et al., 2020:85714). Dengan mengolah data di dekat sumbernya, edge computing mampu memberi respons real-time, menjaga keamanan serta privasi data lokal, dan mendukung pertumbuhan perangkat IoT secara lebih efisien (Cao et al., 2020:85716).

Dari penjelasan di atas, *edge computing* menjadi solusi strategis untuk mempercepat pemrosesan data di lapangan sekaligus mengurangi ketergantungan pada pusat komputasi yang jauh dari sumber data.

b. *Edge AI*

Edge AI merujuk pada penerapan kecerdasan buatan di tepi jaringan, yakni memindahkan fungsi deep learning dan analitik cerdas dari *server* cloud yang jauh ke perangkat edge (Liu et al., 2019:1556). Secara tradisional, model AI yang kompleks dijalankan di pusat data karena membutuhkan sumber daya besar. Untuk menekan latensi dan mendekatkan pemrosesan ke sumber data, fungsi *deep learning* kini dipindahkan ke perangkat *edge* (Liu et al., 2019:1556). Keterbatasan komputasi perangkat edge didukung dengan kerangka perangkat lunak ringan seperti TensorFlow Lite atau Caffe2 yang meminimalkan latensi dan penggunaan memori (Liu et al., 2019:1557). Dukungan akselerator perangkat keras, misalnya Edge TPU, juga memungkinkan algoritma machine learning dieksekusi secara efisien dengan konsumsi daya rendah (Liu et al., 2019:1558)

Dengan uraian tersebut, *Edge AI* memungkinkan pemrosesan cerdas berlangsung di dekat sumber data sehingga respons lebih cepat, privasi lebih terjaga, dan penggunaan sumber daya tetap efisien.

c. *Internet of Things* (IoT)

Menurut Atzori et al., dalam (Agustini, 2025:37), *Internet of Things* (IoT) telah berkembang pesat dan memberikan dampak signifikan pada sektor manufaktur, kesehatan, transportasi, serta *smart city*. Secara konsep, IoT memungkinkan berbagai perangkat untuk saling berkomunikasi dan bertukar data secara *real-time*, sehingga

menciptakan ekosistem yang lebih otomatis dan efisien (Agustini, 2025:37) Agustini. Meskipun IoT menawarkan lompatan besar dalam otomatisasi, implementasinya di dunia nyata masih dihadapkan pada beberapa tantangan krusial. Menurut Weber dalam Agustini (2025:27), tantangan utama implementasi IoT meliputi latensi, penggunaan pita lebar (bandwidth) yang tinggi, serta masalah keamanan data.

3. Kecerdasan Buatan dan Computer Vision

Penerapan kecerdasan buatan pada pemrosesan citra digital memperluas kemampuan rekayasa perangkat lunak untuk meniru penglihatan manusia. Obi et al. dalam (Setiawati et al., 2025:9) menekankan bahwa computer vision berperan penting dalam mengekstrak informasi dari gambar, sehingga mesin dapat menafsirkan data visual secara otomatis. Sejalan dengan itu, Flores-Fuentes et al. dalam (Ardiansyah et al., 2025:1152) menjelaskan bahwa teknologi ini meniru cara manusia mengenali objek dan memahami konteks lingkungan. Dalam arsitektur pelacakan keberadaan dosen yang otonom dan tahan gangguan visual, Zahra et al. dalam Meldyantono dan Poetro (2024:996) menyatakan bahwa pengenalan wajah dapat menggantikan metode absensi tradisional yang rawan kesalahan dan manipulasi.. Integrasi instrumen cerdas tersebut memungkinkan sistem mengenali identitas dosen melalui representasi vektor wajah yang lebih presisi, sehingga dapat mengurangi kesalahan pencatatan identitas dosen (Meldyantono & Poetro, 2025:997).

Secara keseluruhan, *computer vision* dan pengenalan wajah menjadi fondasi teknis yang memungkinkan sistem melacak keberadaan dosen secara otomatis, akurat, dan andal

a. Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan Buatan (Artificial Intelligence/AI) mencakup subkategori utama seperti Machine Learning, yang berfokus pada pengembangan algoritma analitis. Mitchell dalam Setiawati et al. (2025:9) menjelaskan bahwa Machine Learning memungkinkan komputer belajar dari data empiris untuk membuat prediksi atau keputusan secara otomatis. Pengembangan lebih lanjut dari pendekatan ini adalah Deep Learning. Menurut LeCun et al. dan Schmidhuber dalam Setiawati (2025:9), Deep Learning memanfaatkan jaringan saraf tiruan berlapis (multiple layer) untuk mengolah data berskala besar dan menyelesaikan tugas kompleks, seperti deteksi objek dan pengenalan suara.

Berpijak pada uraian itu, *Machine Learning* dan *Deep Learning* menjadi landasan AI yang memungkinkan sistem belajar dari data serta menangani pemrosesan visual maupun audio secara otomatis dan akurat

b. *Computer Vision*

Computer Vision merupakan pilar utama kecerdasan buatan yang berfokus pada kemampuan mesin mengolah citra visual. Flores-Fuentes et al. dalam Ardiansyah et al. (2025:1152) menjelaskan bahwa

computer vision menganalisis dan menafsirkan data visual secara otomatis sehingga perangkat lunak dapat memahami konteks objek sebagaimana mata manusia. Pemanfaatannya melalui pengenalan wajah otomatis kini menjadi solusi yang relevan. Dony dan Lubis dalam (Terampe & Pramudwiatmoko, 2025:109) menegaskan bahwa sistem berbasis *computer vision* dengan pengenalan wajah menjadi alternatif bagi sistem kehadiran yang sebelumnya dilakukan secara manual. Dalam penelitian ini, kemampuan otentikasi tersebut menjadi fondasi Dosen *Tracker* untuk mengidentifikasi dosen di zona kamera dan memperbarui informasi *last seen location*.

Oleh karena itu, *computer vision* tidak hanya memperkuat otomatisasi pengenalan identitas, tetapi juga menjadi dasar teknis yang memungkinkan Dosen *Tracker* memetakan keberadaan dosen secara akurat di lingkungan kampus.

4. Sistem Pengenalan Biometrik Wajah

Pengenalan biometrik wajah kini menjadi kebutuhan penting dalam sistem otentikasi untuk mendukung keselamatan dan keamanan publik (Rusia & Singh, 2023:1). Bolle et al. dalam Rusia dan Singh (2023:2) menjelaskan bahwa sensor biometrik wajah membedakan individu berdasarkan karakteristik atau penampilan fisik. Sistem ini berkembang pesat dan banyak digunakan pada pengawasan cerdas, layanan keuangan, pemantauan keamanan, investigasi forensik, hingga penerbangan sipil karena menawarkan kenyamanan sekaligus keandalan

tinggi (Rusia & Singh, 2023:2). Penerapannya berskala besar pun kian meluas dan sering dilibatkan dalam keputusan kritis, termasuk penegakan hukum (Terhorst et al., 2022:16).

Hal ini menunjukkan bahwa biometrik wajah telah bertransformasi dari sekadar alat identifikasi menjadi instrumen andal yang menopang berbagai sistem keamanan dan pengambilan keputusan di ranah publik

a. Sistem Biometrik Wajah Dasar

Pengenalan wajah berlangsung melalui tahapan terstruktur: deteksi wilayah wajah, penyelarasan dan normalisasi, ekstraksi fitur diskriminan, lalu klasifikasi untuk menentukan identitas (Rusia & Singh, 2023:2). Pada tahap ekstraksi, ciri esensial wajah diubah menjadi vektor satu dimensi. Selanjutnya, pencocokan pola (pattern matching) membandingkan citra masukan dengan data di basis data hingga menghasilkan skor kesamaan (similarity score) (Rusia & Singh, 2022:2-3). Skor tersebut kemudian digunakan untuk memvalidasi identitas individu (Rusia & Singh, 2023:3).

Ringkasnya, pengenalan wajah bekerja sebagai rangkaian sistematis dari deteksi hingga verifikasi identitas berdasarkan tingkat kemiripan fitur yang diekstrak.

b. *Face Detection* dan *Face Recognition*

Face detection manusia pada citra dan menandai lokasinya melalui *bounding boxes* (Feng et al., 2022:1). Perkembangannya telah

beralih dari fitur buatan tangan seperti filter Haar menuju CNN end-to-end yang mengekstrak fitur lebih efektif (Feng et al., 2022:1). Setelah deteksi, proses dilanjutkan dengan *face recognition*, yaitu teknik untuk menyimpulkan identitas seseorang dari gambar, baik untuk identifikasi maupun verifikasi (Rusia & Singh, 2023:6). Namun, di lingkungan yang tidak terkondisi, citra wajah rentan terpengaruh pencahayaan, jarak, dan latar belakang, sehingga bervariasi dalam iluminasi, pose, skala, oklusi, keburaman, maupun distorsi (Feng et al., 2022:3).

Menimbang kondisi tersebut, keberhasilan pengenalan wajah tidak hanya bergantung pada akurasi model, tetapi juga pada kemampuannya beradaptasi terhadap variasi visual di lingkungan nyata

c. *Liveness Detection* (Sistem *Anti-Spoofing*)

Sistem pengenalan wajah memiliki kerentanan terhadap serangan presentasi (*presentation attacks*), di mana pelaku dapat menggunakan fitur wajah tiruan duplikat dalam bentuk foto, video, atau topeng (masker) untuk meniru identitas pengguna saat melewati sistem pengenalan wajah (Abdullakutty et al., 2021:57). Serangan ini akan membantu penyerang untuk menembus sistem keamanan jika sistem tersebut tidak memiliki modul deteksi untuk membedakan antara wajah asli dan palsu Abdullakutty et al., (2021:56). Menurut Hernandez-Ortega dkk., dalam Abdullakutty et al. (2021:56), *Presentation Attack Detection* (PAD) bertugas untuk mengidentifikasi apakah sebuah gambar itu asli atau palsu. *Liveness detection* merupakan pendekatan

dinamis dalam PAD yang sangat bergantung pada informasi temporal untuk memverifikasi keberadaan *liveness* (kehidupan) dari *input* yang disajikan ke sensor wajah Abdullakutty et al., (2021:59). Sistem ini mengonfirmasi *liveness* dengan memproses tanda-tanda vital kehidupan seperti denyut nadi, kedipan mata, gerakan bibir, hingga rotasi kepala pengguna Abdullakutty et al., (2021:59).

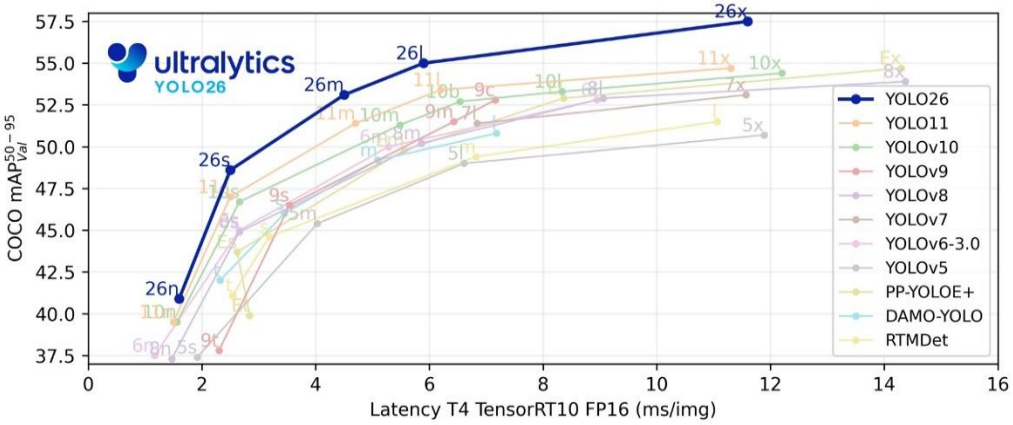
5. Arsitektur Model Deteksi dan Ekstraksi Fitur

Keberhasilan analisis seketika (*real-time*) dalam sistem penglihatan komputer telah bergeser pada arsitektur satu tahap yang efisien, di mana kecepatan inferensi diprioritaskan tanpa mengorbankan akurasi (Chakrabarty, 2026:2). Oleh karena itu, pemisahan proses pembelajaran representasi dari pascapemrosesan heuristik menandai pergeseran fundamental menuju jalur pemrosesan yang dapat dipelajari secara menyeluruh dan lebih sadar terhadap karakteristik perangkat keras Chakrabarty, (2026:24). Selain penemuan lokasi wajah, sistem juga membutuhkan arsitektur jaringan saraf konvolusional mendalam (DCNN) untuk mengubah gambar wajah tersebut menjadi vektor fitur (*embedding*) berdimensi tinggi Terampe & Pramudwiatmoko, (2025:113). Integrasi YOLOv8 untuk deteksi wajah, ArcFace untuk ekstraksi fitur, dan SVM untuk klasifikasi membentuk arsitektur hibrida yang digunakan untuk menjalankan proses pengenalan wajah secara terpaduTerampe & Pramudwiatmoko, (2025:109).

a. Arsitektur Algoritma YOLOv26

Arsitektur YOLO26 mengubah jalur pemrosesan melalui rancangan end-to-end murni dengan penugasan label satu-ke-satu, sehingga modul Non-Maximum Suppression (NMS) tidak lagi diperlukan (Chakrabarty, 2026:7). Pada sisi regresi, YOLO26 juga menghapus Distribution Focal Loss (DFL) dan menyederhanakan dekode kotak distribusi menjadi formulasi yang lebih ringan, ramah perangkat keras, serta mudah diekspor ke format edge seperti ONNX dan TensorRT. Sapkota et al., (2025:4) Penghapusan beban DFL dan prediksi tanpa NMS menghasilkan waktu inferensi yang deterministik, sehingga model ini cocok untuk penerapan instan pada perangkat berdaya rendah (Chakrabarty, 2026:5-7). Proses pelatihan YOLO26 didukung oleh MuSGD untuk meningkatkan stabilitas konvergensi, *Small-Target-Aware Label Assignment* (STAL) untuk menyesuaikan penugasan label terhadap ukuran target, dan *Progressive Loss Balancing* (ProgLoss) untuk mengatur bobot kerugian klasifikasi dan regresi secara dinamis selama pelatihan (Chakrabarty, 2026:9-10).

Dengan demikian, YOLO26 menawarkan deteksi objek yang lebih efisien dan deterministik, sekaligus siap dijalankan pada perangkat *edge* dengan keterbatasan sumber daya.



Gambar 2. 1 *Perbandingan Akurasi dan Kecepatan Inferensi YOLO*

Sumber: chakrabarty. (2026:)

b. Arsitektur ArcFace dan Ekstraksi Fitur

Tahap ekstraksi fitur dilakukan dengan model ArcFace berbasis Deep Convolutional Neural Network (DCNN) bertulang punggung ResNet-100, yang mengubah citra wajah menjadi *embedding* numerik berdimensi 512 (Terampe & Pramudwiatmoko, 2025:113). Keunggulan ArcFace terletak pada fungsi kerugian Additive Angular Margin Loss. Fungsi ini memperbesar jarak sudut antar identitas berbeda (inter-class) sekaligus memperkecil jarak dalam kelas yang sama (intra-class) di ruang fitur. Menurut Terampe dan Pramudwiatmoko (2025:113), penambahan margin tersebut memperkuat daya diskriminatif model dengan membentuk batas keputusan yang lebih tegas antar kelas. Akibatnya, sampel sekelas terklaster lebih rapat, sementara identitas berbeda terpisah lebih jelas.

Dari sini tampak bahwa ArcFace menghasilkan representasi wajah yang lebih tajam dan mudah dibedakan, sehingga proses identifikasi menjadi lebih akurat.

c. Metode Pencocokan *Cosine Similarity*

Cosine similarity merupakan ukuran untuk menentukan tingkat kemiripan dua vektor bukan nol berdasarkan kosinus sudut di antara keduanya. Nilainya dihitung dengan membagi hasil perkalian titik (dot product) kedua vektor dengan hasil kali panjang masing-masing vektor. Nilai cosine similarity berada pada rentang 1 sampai 1; nilai yang semakin mendekati 1 menunjukkan bahwa kedua vektor memiliki arah dan representasi yang semakin serupa. Dalam sistem berbasis *embedding*, pasangan data dengan nilai cosine similarity tertinggi dianggap memiliki kedekatan representasi yang paling besar (Garcia Quevedo & Kuri, 2026:149).

Dengan demikian, cosine similarity dapat digunakan untuk membandingkan vektor *embedding* tanpa menjadikan besar atau panjang vektor sebagai dasar utama pencocokan. Dalam sistem pengenalan wajah, metode ini dapat diterapkan untuk membandingkan arah vektor *embedding* wajah yang terdeteksi dengan vektor wajah yang tersimpan, sehingga nilai kemiripan dapat digunakan sebagai dasar penentuan identitas..

6. Arsitektur Basis Data (Database)

Arsitektur basis data mengatur cara data disimpan dan ditransformasikan dari tingkat logis hingga penyimpanan fisik. (Shahidinejad et al., 2024:2) membagi perancangan basis data ke dalam empat tahap: analisis kebutuhan, perancangan konseptual, perancangan logis, dan perancangan fisik. Perancangan skema logis menggambarkan struktur data beserta batasan integritas agar sistem memperoleh independensi fisik (Candel et al., 2022:3). Meskipun DBMS relasional menawarkan fungsi yang relatif serupa, implementasi dan perilakunya dapat berbeda secara signifikan (Sotiropoulos et al., 2021:1536). Perbedaan tersebut menuntut lapisan abstraksi yang andal untuk menjembatani kode aplikasi dengan penyimpanan fisik, mengurangi kesalahan kueri, dan meningkatkan portabilitas sistem.

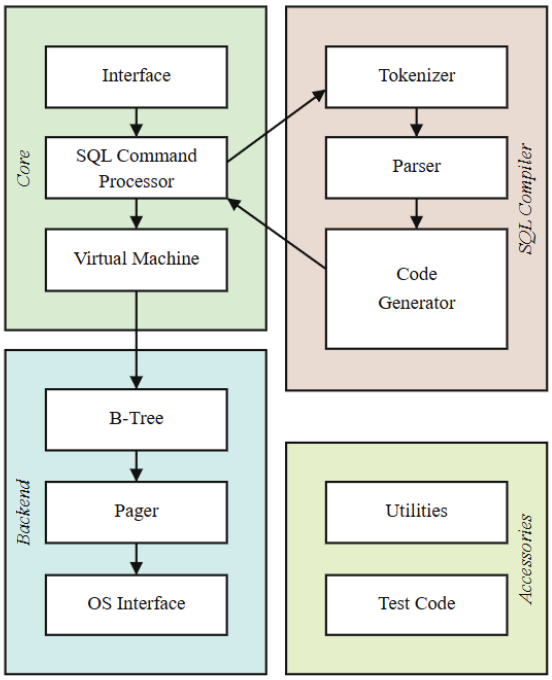
Pada akhirnya, perancangan basis data yang terstruktur dan dilengkapi abstraksi yang tepat menjadi kunci agar pengelolaan data tetap konsisten, fleksibel, dan mudah dikembangkan.

a. Sistem Manajemen Basis Data Lokal (SQLite)

SQLite merupakan sistem manajemen basis data relasional yang tertanam dalam proses aplikasi (in-process). Aplikasi mengelola basis data dengan memanggil fungsi pustaka SQLite secara langsung tanpa memerlukan *server* terpisah. SQLite bersifat lintas platform, ringkas, dan mandiri karena basis data disimpan dalam satu berkas serta tidak membutuhkan dependensi eksternal, instalasi, maupun

konfigurasi khusus. Arsitektur SQLite terdiri atas kelompok modul core, SQL compiler, *backend*, dan accessories. Modul SQL compiler menerjemahkan perintah SQL menjadi bytecode yang dijalankan oleh virtual machine, sedangkan modul *backend* menangani akses halaman basis data dan penyimpanan data melalui sistem operasi (Gaffney et al., 2022: 3535–3537).

Dengan demikian, SQLite dapat digunakan sebagai sistem manajemen basis data lokal yang sederhana dan efisien. Desainnya yang tertanam dalam aplikasi, penyimpanan dalam satu berkas, dan tidak memerlukan konfigurasi *server* menjadikan SQLite sesuai untuk aplikasi yang membutuhkan penyimpanan data lokal secara mandiri.



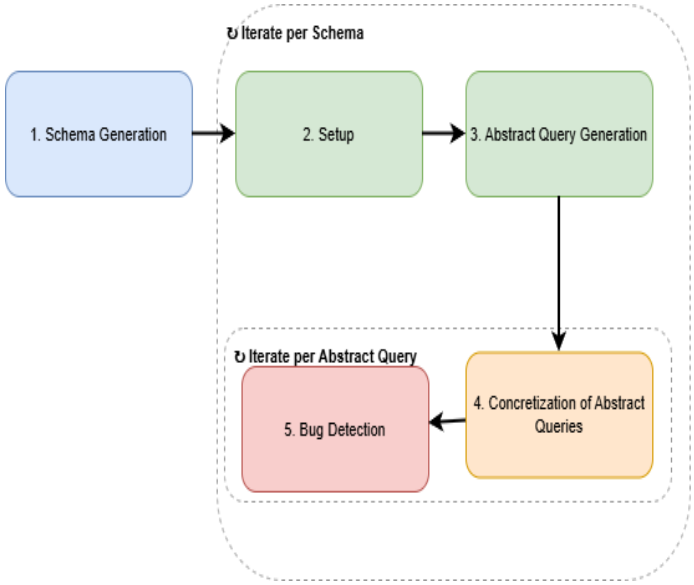
Gambar 2. 2 Arsitektur dari Sqlite

Sumber : (Gaffney et al., 2022)

b. *Object-Relational Mapping* (SQLAlchemy ORM)

bject-Relational Mapping (ORM) adalah teknik pemrograman yang menjembatani perbedaan paradigma antara pemrograman berorientasi objek dan basis data relasional, atau *impedance mismatch* (Sotiropoulos et al., 2021:1535). ORM menyediakan antarmuka berorientasi objek di atas basis data relasional, sehingga objek program dapat disimpan dan diambil tanpa menulis kueri SQL secara manual. Skema basis data diabstraksikan melalui kelas khusus yang disebut model, sementara rekaman data diwakili oleh instansiasi objek kelas tersebut (Sotiropoulos et al., 2021:1536). SQLAlchemy merupakan salah satu ORM Python yang menyediakan API berorientasi objek untuk menyembunyikan detail mekanis kueri SQL dari pengembang (Sotiropoulos et al., 2021:1536). Konsep pemetaan dua arah antara objek kelas dan tabel relasional digambarkan pada Gambar 2.3, yang mempermudah pemahaman alur kerja ORM secara visual.

Maka dapat disimpulkan bahwa ORM menyederhanakan akses basis data dengan memetakan objek program ke tabel relasional, sehingga pengembangan sistem menjadi lebih rapi, efisien, dan mudah dipelihara.



Gambar *2. 3 Overview of our approach for automatically testing ORMs.*,
Sumber: (Sotiropoulos et al., 2021:1538)

7. Kerangka Kerja (*Framework*) Perangkat Lunak

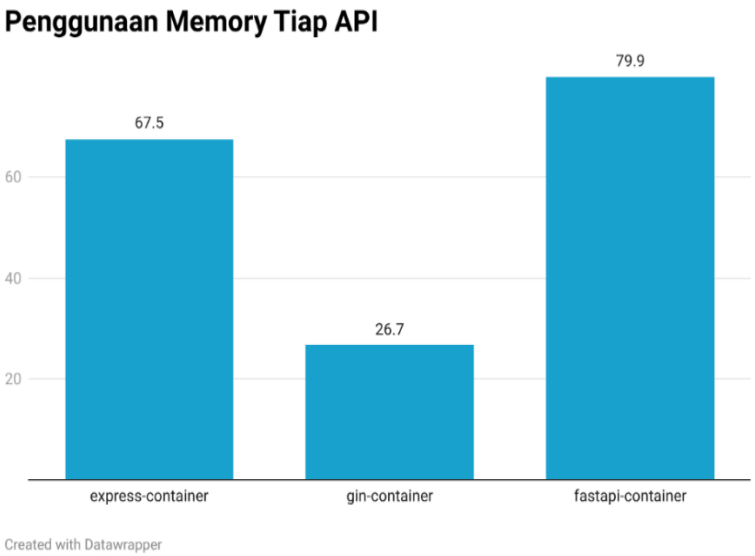
Pengembangan perangkat lunak modern menuntut efisiensi waktu, pengelolaan basis kode yang terstruktur, dan skalabilitas tinggi. Penulisan kode dari nol tanpa panduan arsitektur sudah kurang relevan karena berisiko menimbulkan tumpang tindih logika. Kerangka kerja perangkat lunak hadir sebagai fondasi yang menyediakan komponen bawaan untuk mempercepat siklus rekayasa. Integrasi kerangka kerja memudahkan pertukaran data, fungsionalitas, dan fitur antar-layanan sehingga menghemat waktu dan tenaga pengembang Nguyen et al., 2019, sebagaimana dikutip dalam (Santiago & Benisius, 2025:1220). Selain itu, arsitektur asinkron pada *server* component terbukti efisien dalam menekan beban rendering halaman *web* dinamis Sinulingga & Suartana, (2024:539-541). Oleh karena itu, penelitian ini memadukan infrastruktur tersebut.

Dengan uraian di atas, kerangka kerja dan arsitektur asinkron menjadi fondasi penting untuk membangun sistem yang lebih cepat dikembangkan, terstruktur, dan efisien dalam pengelolaan beban.

a. *Framework Backend* (FastAPI)

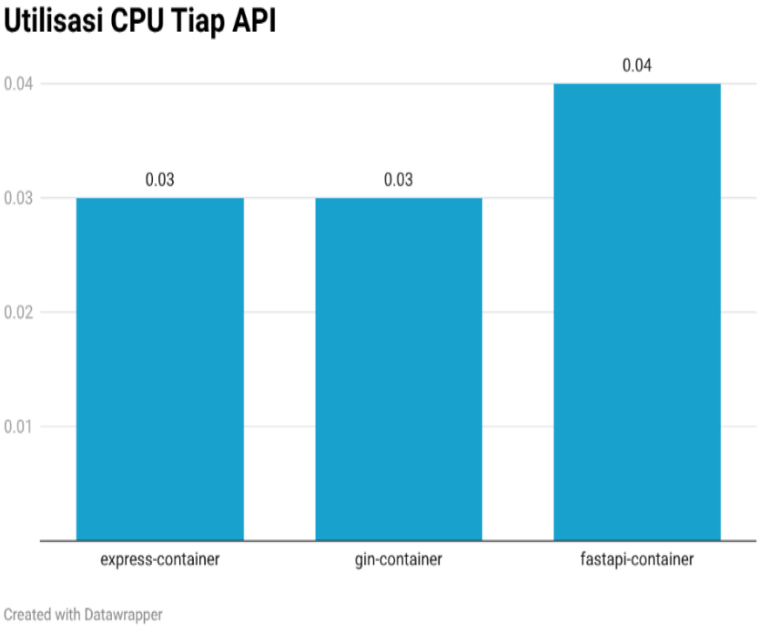
FastAPI adalah kerangka kerja *backend* berbasis Python yang dirancang untuk mengeksekusi instruksi data berkinerja tinggi melalui pendekatan asinkron (Asynchronous Server Gateway Inter*face*) Raihan et al., 2022, sebagaimana dikutip dalam (Suryotomo et al., 2024:114). Pengujian beban oleh Santiago dan Benisius (2025:1225) menunjukkan utilisasi CPU rata-rata 0,04% dan penggunaan memori rata-rata 79,90 MB. Kerangka kerja ini efisien untuk membangun API yang cepat dan andal, serta secara otomatis menyediakan validasi data dan dokumentasi terintegrasi (Kornienko et al., 2021, sebagaimana dikutip dalam Wahyi & Jamzuri, 2023:2). Pendekatan asinkronnya memungkinkan komunikasi data tetap responsif tanpa menunggu permintaan sebelumnya selesai dieksekusi Azhari, 2022, sebagaimana dikutip dalam (Wahyi & Jamzuri, 2023:2).

Berdasarkan temuan tersebut, FastAPI menjadi pilihan *backend* yang tepat untuk sistem yang menuntut respons cepat, efisiensi sumber daya, dan kemudahan pengembangan API.



Gambar *2. 4* Penggunaan memori tiap API

(Sumber: Santiago & Benisius, 2025:1224)



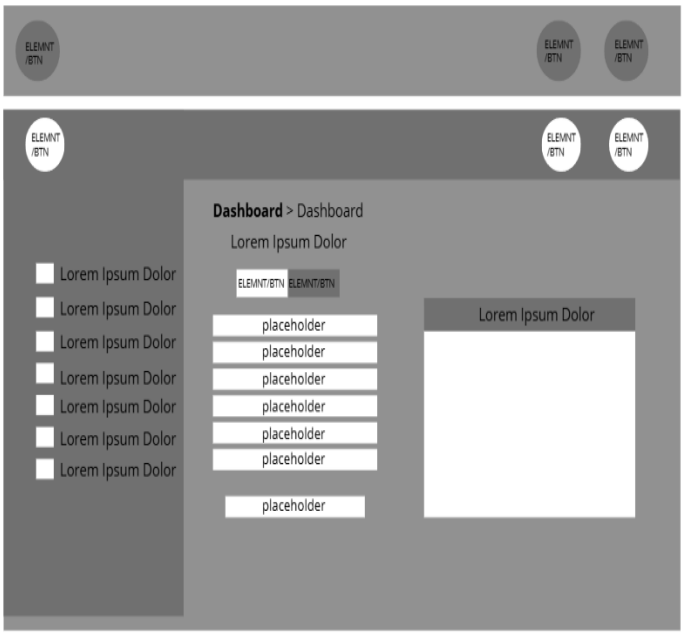
Gambar *2. 5* Utilisasi CPU tiap API,

(Sumber: Santiago & Benisius, 2025:1224)

b. *Framework Frontend* (React.js)

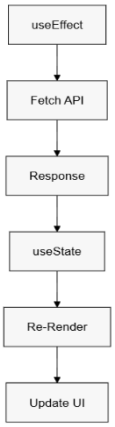
React.js adalah pustaka JavaScript sumber terbuka untuk membangun antarmuka interaktif pada aplikasi halaman tunggal (Single Page Application). Data mentah dari *server* diubah menjadi tampilan visual melalui Document Object Model (DOM) di peramban Nordström & Dixelius, 2023, sebagaimana dikutip dalam (Sinulingga & Suartana, 2024:533). Setiap komponen React mengelola bagian antarmuka secara spesifik, tetapi tetap terintegrasi dalam kerangka Virtual DOM. Sinkronisasi status dinamis ini memungkinkan pembaruan visual misalnya perubahan lokasi terakhir dosen tampil secara instan, terutama bila didukung *Server-Side Rendering* untuk menjaga performa situs (Lyxell, 2022, sebagaimana dikutip dalam Sinulingga & Suartana, 2024:535). Stabilitas manipulasi data tersebut menghasilkan navigasi yang mulus bagi pengguna, khususnya mahasiswa yang memantau status keberadaan dosen, dan contoh gambaran dari react *dashboard* dan ilustrasi react hooks yang bisa di lihat pada gambar 2.6 dan gambar 2.7.

Dengan demikian, React.js mendukung penyajian informasi lokasi dosen secara responsif dan nyaman diakses melalui antarmuka *web* yang cepat dan interaktif.



Gambar 2. 6 Halaman Dashboard Dari react

Sinulingga & Suartana, (2024)



Gambar 2. 7 *Fetching data pada React Hooks,*

Sumber: Sinulingga, (2024)

8. Protokol Komunikasi dan Transmisi Waktu Nyata

Sistem Dosen *Tracker* di ekosistem komputasi tepi membutuhkan infrastruktur jaringan yang stabil dan responsif. Pembaruan data secara manual dinilai kurang efektif karena berpotensi membebani memori

peladen. Karena itu, protokol komunikasi waktu nyata menjadi landasan penting untuk menjembatani pertukaran informasi seketika antara perangkat perekam, modul kecerdasan buatan, dan kanal distribusi informasi (Irawan & Suartana, 2024:138). Protokol jaringan tersebut dirancang terpadu agar mampu menyalurkan tayangan video secara kontinu sekaligus menyebarkan notifikasi lokasi terakhir dosen ke gawai mahasiswa. Integrasi ini menjaga sinkronisasi seluruh lapisan aplikasi tetap aktual dan minim penundaan.

Secara keseluruhan, infrastruktur jaringan waktu nyata menjadi penopang utama agar deteksi, pemrosesan, dan penyampaian lokasi dosen berjalan selaras tanpa keterlambatan

a. *Webhook* dan *Server-Sent Events* (SSE)

Webhook adalah metode transmisi berbasis kejadian (*event*-driven) yang mengirimkan panggilan balik otomatis ke platform pihak ketiga begitu aksi tervalidasi (Wijianto, 2025:28). Sementara itu, Server-Sent Events (SSE) merupakan aliran data searah di mana peladen terus mengirim pembaruan ke peramban tanpa diminta ulang (Irawan & Suartana, 2024:138). SSE berfungsi sebagai saluran ringan yang menekan jeda jaringan saat pemuatan riwayat interaksi pengguna. Masing-masing saluran notifikasi menjaga efisiensi transmisi, dan keduanya saling melengkapi dalam membangun komunikasi tanpa beban komputasi berat. Kolaborasi kedua protokol ini memuluskan

penyampaian notifikasi lokasi terakhir dosen ke perangkat mahasiswa dengan konsumsi sumber daya yang minimal.

Maka dapat disimpulkan bahwa *Webhook* dan SSE membentuk jalur komunikasi waktu nyata yang ringan, efisien, dan andal untuk menyebarkan pembaruan lokasi dosen secara massal.

b. *WebSocket* dan *Web Real-Time Communication* (WebRTC)

WebSocket adalah protokol transmisi dua arah penuh (full-duplex) berkinerja tinggi yang menjaga koneksi tetap terbuka antara klien dan peladen (Irawan & Suartana, 2024:140). Saluran ini menekan latensi pengiriman pesan karena tidak memerlukan negosiasi alamat jaringan berulang. Sebagai pendamping, Web Real-Time Communication (WebRTC) memfasilitasi komunikasi media waktu nyata melalui peramban (Limaran et al., 2025:1573). Keduanya berperan dalam komunikasi waktu nyata, tetapi WebRTC dioptimalkan untuk media, sedangkan WebSocket digunakan untuk data interaktif (Limaran et al., 2025:1573). Integrasi tersebut membentuk aliran multimedia yang stabil dan andal, sehingga tampilan kamera langsung di layar pengguna tetap jelas dan minim gangguan visual.

Dari sisi komunikasi, *WebSocket* dan WebRTC menjadi penopang utama agar penyaluran media dan data interaktif berjalan cepat, stabil, dan responsif di lingkungan sistem.

c. *Real-Time Streaming Protocol* (RTSP)

Real-Time Streaming Protocol (RTSP) adalah kerangka jaringan tingkat aplikasi untuk mengatur dan mengendalikan aliran media visual dari sumber perekam ke penerima. Iqbal (2022) dalam (Opitasari et al., 2025:94) RTSP merupakan protokol yang digunakan untuk mengendalikan peladen media *streaming*. Setiap paket video menyuplai informasi optik, tetapi tetap terpadu sebagai pasokan visual utuh tanpa tabrakan data. Protokol ini terus mengalirkan deretan citra ke modul pengenalan wajah agar fitur biometrik dapat diekstraksi secara presisi. Konsistensi transmisi tersebut penting untuk menopang keberhasilan pelacakan identitas dalam ekosistem Dosen *Tracker* yang dinamis.

Berpijak pada uraian itu, RTSP menjadi saluran utama yang menjaga kontinuitas umpan video agar proses pengenalan wajah dan pelacakan dosen tetap akurat dan stabil.

9. Metodologi Rekayasa Perangkat Lunak

Metodologi rekayasa perangkat lunak merupakan kerangka manajerial yang menata alur perancangan, pembuatan, pengujian, dan pemeliharaan sistem informasi. Ketepatan pemilihan metode sangat menentukan pengelolaan waktu, koordinasi tugas, serta kemampuan tim merespons perubahan kebutuhan pengguna. Karena siklus hidup klasik sering terasa kaku, pendekatan modern yang menekankan iterasi lincah dan jaminan kualitas validasi semakin banyak digunakan (Sidiq et al.,

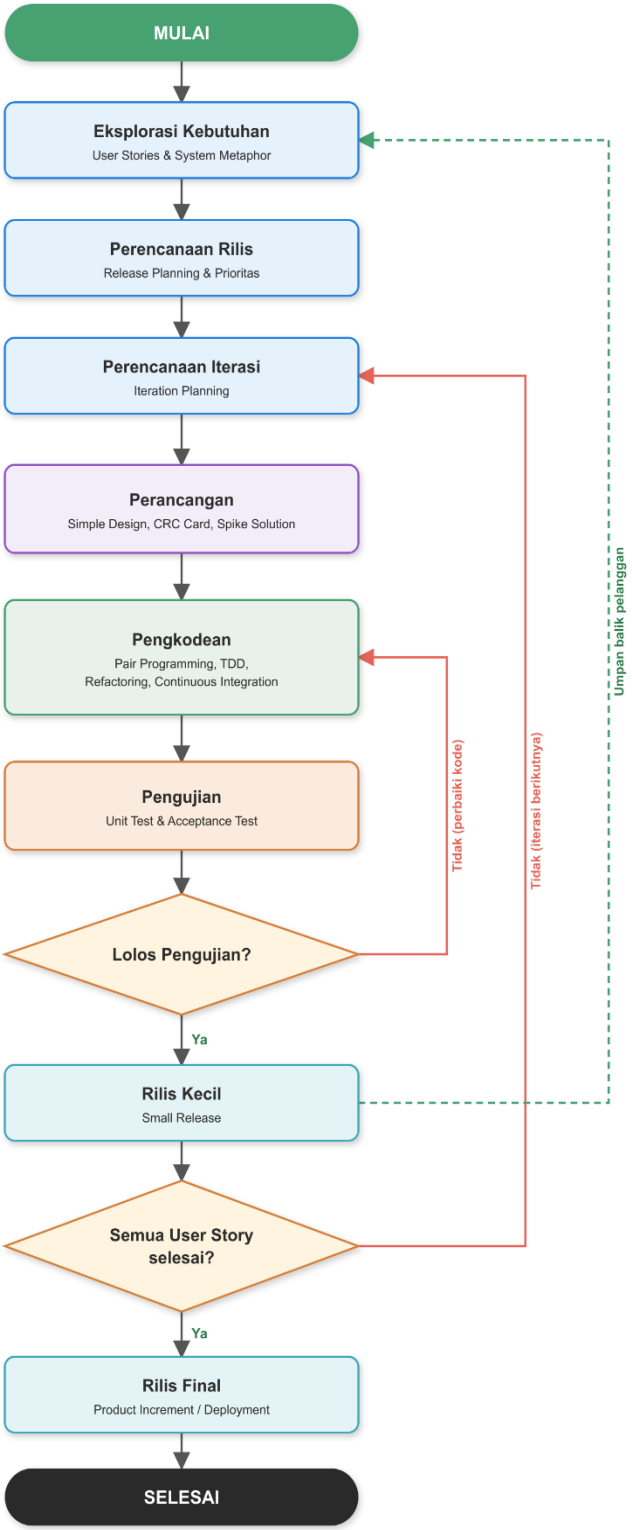
2025:42). Kombinasi metodologi tersebut menjaga fungsionalitas algoritma tetap andal meski spesifikasi teknis terus berubah. Kerja sama pendekatan empiris ini pun menjadi penopang rilis produk yang stabil, tangguh, dan mudah dikembangkan lebih lanjut.

Pada akhirnya, keberhasilan pengembangan sistem sangat bergantung pada metodologi yang terstruktur sekaligus adaptif terhadap dinamika kebutuhan teknis dan pengguna.

a. *Extreme Programming* (XP)

Extreme Programming (XP) adalah pendekatan Agile yang dirancang untuk merespons perubahan kebutuhan sistem melalui iterasi perilisan yang pendek (Sidiq et al., 2025:42). Metodologi ini menekankan kolaborasi intensif antara pengembang dan perwakilan pengguna (Suwondo et al., 2023:2). Setiap tahap dari perencanaan hingga pelepasan fitur menjaga komunikasi teknis tetap utuh dalam ritme pengembangan yang teratur. Dengan pola kerja tersebut, penyempurnaan fungsi pelacakan lokasi dan distribusi notifikasi dapat dievaluasi serta diimplementasikan secara berkesinambungan, sehingga tumpukan perubahan modul yang mendadak dapat dihindari.dadak.

Dengan demikian, XP mendukung pengembangan Dosen *Tracker* yang lebih adaptif, terarah, dan responsif terhadap perubahan kebutuhan pengguna. Untuk lebih jelasanya alur bisa di lihat di gambar 2.8 berikut



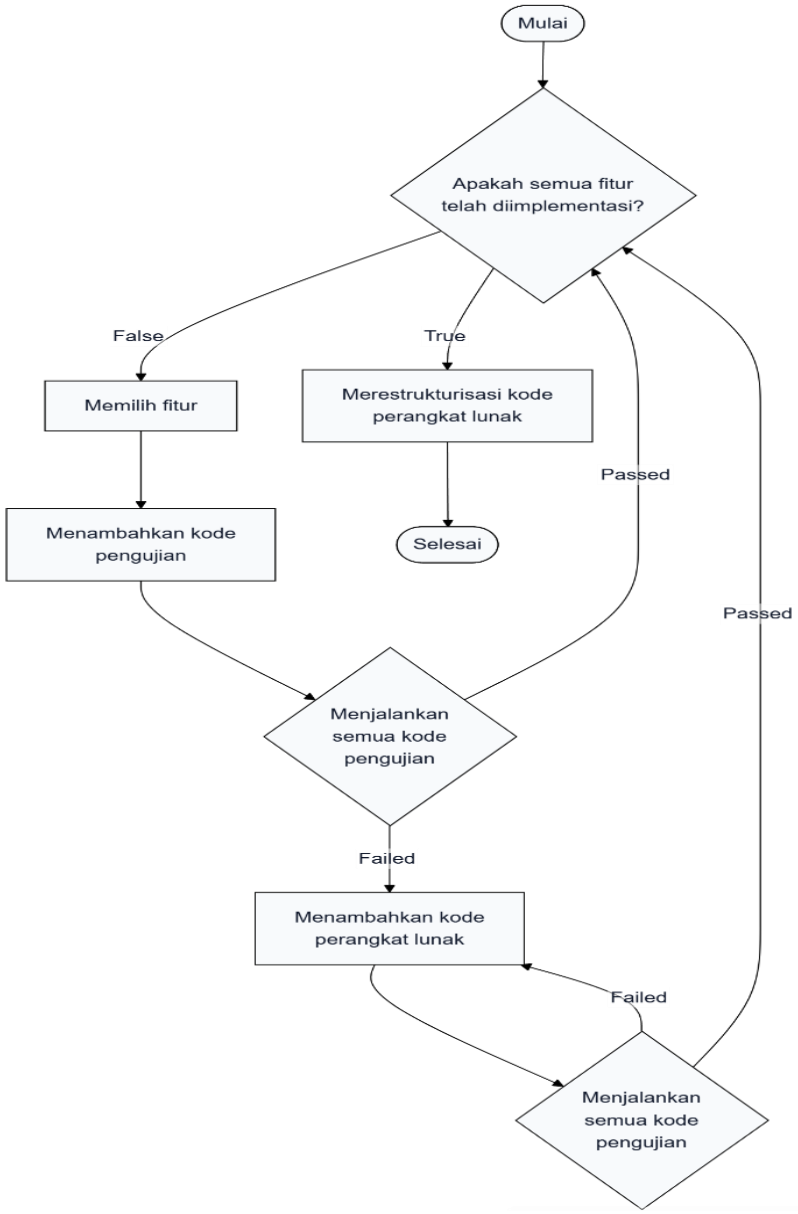
Gambar 2. 8 Kartu *Class Responsibilities Collaboration* (CRC)

Sumber: Suwondo et al., (2023)

b. *Test-Driven Development* (TDD)

Test-Driven Development (TDD) adalah strategi pengembangan perangkat lunak di mana skenario pengujian unit disusun terlebih dahulu sebelum kode fungsional ditulis (Agustin et al., 2026:6). Siklusnya mewajibkan kode melalui tahapan gagal, lulus, lalu disederhanakan secara berkelanjutan. Setiap pengujian otomatis memvalidasi keluaran fungsi tertentu dan saling terintegrasi menjaga arsitektur peladen tanpa benturan logika. Pendekatan ini menuntun pengembang tetap berpegang pada spesifikasi awal, sehingga kode yang dihasilkan lebih modular dan penambahan modul baru tidak merusak kestabilan sistem. TDD menghasilkan kode yang lebih andal serta mendukung pengembangan fitur baru secara iteratif dan teruji dengan baik (Wijanarko & Subhiyanto, 2024:63) serta memangkas waktu perbaikan di kemudian hari. Dalam arsitektur Dosen *Tracker*, pengujian berlapis ini penting untuk memastikan alur deteksi, pembaruan *last seen location*, dan pengiriman notifikasi Telegram berjalan tanpa celah fungsional. Alur siklus TDD digambarkan pada Gambar 2.9 (Agustin et al., 2026:5).

Hal ini menegaskan bahwa TDD menjadi mekanisme penjamin mutu yang menjaga keandalan sistem sejak tahap awal pengembangan hingga operasional.



Gambar 2. 9 Bagan Alir Siklus TDD

Sumber: Agustin et al., (2026:5)

10. Pemodelan Visual Perangkat Lunak (UML)

Pemodelan visual perangkat lunak merupakan tahap penting dalam perancangan sistem karena berfungsi menggambarkan rancangan perangkat lunak secara terstruktur dan mudah dipahami sebelum masuk ke

tahap implementasi. Melalui pemodelan visual, kebutuhan pengguna, alur proses, serta struktur data dapat dikomunikasikan secara jelas kepada seluruh pihak yang terlibat dalam pengembangan. Salah satu standar pemodelan yang paling banyak digunakan adalah *Unified Modelling Language* (UML), yaitu notasi tujuan umum yang menyediakan seperangkat mekanisme untuk memodelkan struktur, perilaku, dan interaksi sebuah sistem perangkat lunak (Luhukay & Mailoa, 2024 dikutip dalam Devina & Rosmintauli, (2025:11). UML terdiri atas beragam diagram yang saling melengkapi untuk merepresentasikan sistem dari berbagai sudut pandang. Dalam penelitian ini, UML digunakan untuk memodelkan sistem Dosen *Tracker* berbasis pengenalan wajah melalui flowchart dan alur sistem, *use case* diagram, pemodelan arsitektur objek (*class diagram*), *activity diagram*, serta *sequence* diagram, sehingga rancangan sistem dapat dianalisis dan diverifikasi secara menyeluruh sebelum dibangun.

a. *Flowchart* dan Alur Sistem

Flowchart merupakan representasi diagramatis dari suatu proses atau alur kerja yang digunakan untuk memvisualisasikan langkah-langkah dalam suatu sistem secara sistematis sehingga memudahkan pemahaman terhadap proses tersebut Khesya, 2021 dikutip dalam (Malik, 2025:17). Dengan menggunakan simbol-simbol baku seperti *terminator*, *input/output*, *process*, dan *decision*, *flowchart* mampu menjelaskan urutan kegiatan beserta percabangan

logika yang terjadi di dalam sistem. Penggunaan *flowchart* sangat membantu dalam menganalisis alur sebelum sistem diprogram, karena kesalahan logika dapat terlihat lebih awal pada tingkat perancangan. Dalam penelitian ini, flowchart dipakai untuk menggambarkan alur proses pelacakan keberadaan dosen berbasis pengenalan wajah, mulai dari pengambilan citra oleh kamera, proses pendeteksian dan pengenalan wajah, pencocokan dengan data yang tersimpan, pembaruan *last seen location* berdasarkan zona kamera, hingga distribusi notifikasi lokasi melalui *bot* Telegram. Melalui flowchart ini, keseluruhan alur kerja sistem menjadi lebih mudah dipahami dan ditelusuri apabila terjadi kesalahan pada salah satu tahap.

Tabel 2.1 Simbol dan Fungsi Flowchart

Simbol	Nama	Fungsi
	*Terminator*	Menandakan titik awal (start) atau titik akhir (end) dari berjalannya sebuah program.
	*Flow* (Garis Alir)	Arah atau urutan jalannya proses.
	*OnPage Connector*	Penghubung alur di halaman yang sama.

Simbol	Nama	Fungsi
	*Off Page Connector*	Penghubung alur ke halaman yang berbeda.
	*Input/Output*	Proses memasukkan data atau menampilkan hasil.
	*Process*	Tindakan atau proses pengolahan data.
	*Decision*	Pengambilan keputusan atau percabangan alur (ya/tidak).

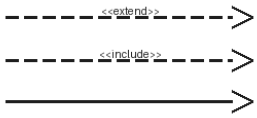
Sumber: Keysha dalam malik (2025)

b. *Use Case Diagram*

Use case diagram merupakan model diagram UML yang digunakan untuk menggambarkan kebutuhan fungsional (*requirement*) yang diharapkan dari sebuah sistem, yaitu menggambarkan secara ringkas siapa yang menggunakan sistem dan apa saja yang dapat dilakukannya (Dharmawan, 2023 dikutip dalam Devina et al., 2025:12). Diagram ini menampilkan hubungan antara aktor sebagai pengguna sistem dengan sejumlah *use case* yang merepresentasikan fungsi atau layanan yang disediakan sistem. Dengan demikian, *use case diagram* membantu analis dalam menyusun dan menyepakati kebutuhan sistem sejak tahap awal pengembangan. Dalam penelitian ini, *use case* diagram menggambarkan interaksi antara beberapa aktor, yaitu mahasiswa,

admin, dan sistem, dengan berbagai fungsi Dosen *Tracker* berbasis pengenalan wajah seperti mendeteksi dan mengidentifikasi dosen, memperbarui *last seen location*, mengelola data dosen, mengelola zona kamera, serta menerima atau melihat informasi lokasi terakhir dosen melalui *bot* Telegram maupun antarmuka *web*. Pemetaan ini memastikan seluruh kebutuhan fungsional utama telah tercakup dan sesuai dengan peran masing-masing pengguna, sehingga rancangan sistem lebih terarah dan mudah diverifikasi.

Tabel 2. 2 Simbol *Use Case Diagram*

Simbol	Nama	Fungsi
	*Association*	Menandakan titik awal (start) atau titik akhir (end) dari berjalannya sebuah program.
	*Actor*	Merepresentasikan entitas di luar sistem (Dosen/*Admin*) yang berinteraksi langsung dengan sistem.
	*Use Case*	Dalam pemodelannya, *use case* direpresentasikan dengan simbol oval yang mencantumkan nama skenario atau fungsi sistem.

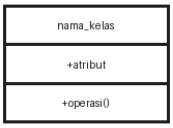
Sumber: Devina (2025)

c. Pemodelan Arsitektur Objek

Pemodelan arsitektur objek digambarkan melalui *class diagram*, yaitu diagram yang menunjukkan kelas-kelas (*class*) dalam sistem beserta hubungannya secara logis (Revita Elinda et al., 2023 dikutip dalam Devina et al., 2025:12). *Class diagram* menyajikan deskripsi lengkap dari kelas-kelas yang ditangani sistem, di mana setiap kelas dilengkapi dengan atribut dan operasi (metode) yang diperlukan, sehingga struktur statis sistem dapat dipahami secara utuh. Melalui diagram ini, relasi antar kelas seperti asosiasi, agregasi, maupun pewarisan dapat terlihat jelas dan sekaligus menjadi dasar dalam perancangan basis data serta implementasi kode berbasis objek. Dalam penelitian ini, *class diagram* menggambarkan arsitektur objek dari sistem Dosen *Tracker* berbasis pengenalan wajah, mencakup kelas-kelas seperti data dosen, data wajah (*face embedding*), zona kamera, dan log *last seen location*, beserta hubungan antar kelas tersebut. Dengan demikian, *class diagram* memudahkan pengembang memahami bagaimana data lokasi disimpan, diproses, dan saling berinteraksi di dalam sistem.

Tabel 2. 3 Simbol *Class Diagram*

Simbol	Nama	Fungsi
	Association	Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan multiplicity.

	Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	Agregasi / aggregation	Relasi antar kelas dengan makna semuabagian (whole-part).
	Generalization	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
	Antarmuka / inter*face*	Sama dengan konsep inter*face* dalam pemrograman berorientasi objek.
	Kebergantungan / dependency	Relasi antar kelas dengan makna kebergantungan antar kelas.

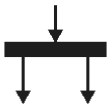
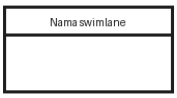
Sumber: Devina (2025)

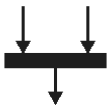
d. *Activity Diagram*

Activity diagram merupakan rancangan aliran aktivitas atau aliran kerja dalam sebuah sistem yang akan dijalankan (Dharmawan, 2023 dikutip dalam Devina et al., 2025:12). Diagram ini dirancang berdasarkan satu atau beberapa *use case* dan merepresentasikan proses yang berjalan pada sistem, termasuk urutan aktivitas, percabangan keputusan, serta kegiatan yang dilakukan secara paralel. Berbeda dengan *use case* yang menekankan pada apa yang dilakukan aktor, *activity diagram* lebih menekankan pada bagaimana proses berlangsung dari awal hingga akhir. Dalam penelitian ini, *activity*

diagram digunakan untuk menggambarkan alur aktivitas saat dosen terdeteksi pada zona kamera melalui pengenalan wajah, mulai dari kamera menangkap citra, sistem memverifikasi identitas, pemeriksaan keaslian wajah (*anti-spoofing*), hingga sistem memperbarui atau menolak pembaruan *last seen location* berdasarkan hasil verifikasi, lalu mengirimkan notifikasi lokasi kepada mahasiswa melalui *bot* Telegram. Dengan menggambarkan setiap langkah beserta kondisinya, *activity diagram* memudahkan penelusuran logika proses dan memastikan tidak ada tahap penting yang terlewat dalam alur kerja sistem.

Tabel 2. 4 Simbol *Activity* Diagram

Simbol	Nama	Fungsi
	Aktivitas	Menggambarkan aktivitas/kegiatan yang dilakukan oleh sistem, biasanya diawali dengan kata kerja.
	*Fork*	Percabangan yang digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel (satu masuk, banyak keluar).
	Status Akhir	Menunjukkan status akhir dari sebuah diagram aktivitas.
	*Decision*	Menunjukkan kondisi di mana terdapat pilihan aktivitas yang lebih dari satu.
	*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Simbol	Nama	Fungsi
	*Join*	Penggabungan/rake yang digunakan untuk menunjukkan adanya dekomposisi (banyak masuk, satu keluar).
	Status Awal	Menunjukkan status awal dari sebuah diagram aktivitas.
	Penggabungan	Di mana lebih dari satu aktivitas lalu digabungkan menjadi satu.

Sumber: Devina (2025)

e. *Sequence Diagram*

Sequence diagram merupakan kolaborasi yang dinamis antar beberapa objek yang memperlihatkan rangkaian pesan yang dikirimkan antar objek serta interaksi yang terjadi antar objek dalam sistem yang dibangun (Noviantoro et al., 2022 dikutip dalam Devina et al., 2025:13). Diagram ini menekankan pada dimensi waktu, yaitu urutan pesan yang saling dipertukarkan antar objek untuk menyelesaikan suatu skenario tertentu. Melalui *sequence diagram*, dapat terlihat objek mana yang aktif, pesan apa yang dikirim, serta respons yang diberikan pada setiap tahap interaksi, sehingga alur komunikasi internal sistem menjadi lebih jelas. Dalam penelitian ini, *sequence* diagram menggambarkan urutan interaksi antara kamera atau edge node, modul pengenalan wajah, *server*, dan *bot* Telegram saat proses pelacakan lokasi berlangsung, mulai dari pengiriman citra, permintaan verifikasi, pencocokan data, pembaruan *last seen location*, hingga pengiriman

notifikasi status keberadaan dosen. Diagram ini melengkapi *activity diagram* dengan memperlihatkan bagaimana antar komponen sistem saling berkomunikasi secara berurutan.

Tabel 2. 5 Simbol Squence diagram

Simbol	Nama	Fungsi
	*Activation*	Menunjukkan periode ketika suatu objek sedang menjalankan proses atau menerima kendali eksekusi.
	*Actor*	Mewakili pengguna atau pihak luar yang berkomunikasi dengan sistem selama proses berlangsung.
	Lifeline	Menggambarkan keberadaan suatu objek selama interaksi berlangsung dalam *sequence* diagram.
	*Message*	Menandakan pengiriman informasi atau permintaan dari satu objek menuju objek lainnya.
	*Entity Class*	Merepresentasikan objek yang menyimpan atau membawa data yang digunakan dalam sistem.
	*Recursive Message*	Memperlihatkan pemanggilan operasi oleh objek terhadap dirinya sendiri.
	*Boundary Class*	Menggambarkan komponen antarmuka yang menjadi penghubung antara pengguna dengan sistem.

	*Control Class*	Berfungsi mengatur alur proses serta mengoordinasikan komunikasi antara antarmuka dan data.

Sumber: Devina (2025)

11. Metrik Evaluasi Sistem

Metrik evaluasi sistem digunakan untuk mengukur kemampuan model pembelajaran mesin dalam menghasilkan prediksi pada data baru yang belum pernah digunakan selama pelatihan. Evaluasi menggunakan data yang berbeda dari data pelatihan diperlukan agar hasilnya dapat memperkirakan kemampuan generalisasi model secara objektif (Varoquaux & Colliot, 2026:602). Pemilihan metrik perlu disesuaikan dengan karakteristik data dan tujuan penerapan karena setiap metrik menggambarkan aspek kinerja yang berbeda. Penggunaan akurasi saja dapat menghasilkan kesimpulan yang menyesatkan pada data tidak seimbang, sebab model yang selalu memprediksi kelas mayoritas tetap dapat memperoleh nilai akurasi yang tinggi (Varoquaux & Colliot, 2026:605).

Dengan demikian, evaluasi sistem tidak cukup dilakukan menggunakan satu metrik. Beberapa metrik perlu digunakan secara bersama untuk mengetahui kemampuan model dalam menghasilkan prediksi yang benar, mengenali setiap kelas, serta membedakan jenis kesalahan yang dihasilkan.

a. *Confusion Matrix* dan *Precision-Recall*

Confusion matrix merupakan matriks yang merangkum perbandingan antara kelas aktual dan kelas hasil prediksi. Pada klasifikasi biner, hasil prediksi dibagi menjadi true positive, true negative, false positive, dan false negative (Varoquaux & Colliot, 2026:603–604). Komponen tersebut digunakan untuk menghitung precision dan recall. Precision menunjukkan proporsi sampel yang benar-benar positif dari seluruh sampel yang diprediksi positif, sedangkan *recall* menunjukkan proporsi sampel positif yang berhasil dikenali dari seluruh sampel yang sebenarnya positif(Varouquaux & Colliot, 2026:604). Kurva Precision–Recall menggambarkan hubungan antara kedua metrik pada berbagai nilai ambang dan berfokus pada kemampuan model dalam mengenali kelas minoritas (Varoquaux & Colliot, 2026:608-609).

Secara ringkas, confusion matrix memberikan informasi mengenai jenis prediksi benar dan salah yang dihasilkan model, sedangkan precision, recall, dan kurva Precision–Recall memberikan penilaian yang lebih terarah terhadap kelas positif. Kombinasi metrik tersebut membuat evaluasi kinerja sistem lebih menyeluruh, khususnya ketika distribusi kelas tidak seimbang.

12. Layanan Pihak Ketiga Terintegrasi

Layanan pihak ketiga terintegrasi merupakan komponen penting dalam arsitektur teknologi informasi untuk memperluas jangkauan

operasional sistem secara efisien. Integrasi berbasis pesan instan dinilai efektif meredam hambatan psikologis pengguna saat mengadopsi sistem baru. (Widya & Airlangga, 2020:14) menjelaskan bahwa tingginya penggunaan aplikasi pesan instan membuat masyarakat sudah terbiasa dengan platform tersebut, sehingga kelemahan aplikasi mobile mandiri yang menuntut pembelajaran antarmuka baru dapat diatasi. Dari sisi teknis, akses lintas platform juga membuat integrasi ini diminati administrator. Ridho dalam (Andika et al., 2022:128) menyatakan bahwa Telegram menguntungkan karena dapat berjalan di hampir semua gawai dan memudahkan pembangunan notifikasi otomatis melalui API terbuka. Efisiensi integrasi semakin diperkuat dukungan kode asinkron pada *server bot* modern. Menurut (Belson, 2024:2), pola async/await menyederhanakan pemrograman asinkron dengan memungkinkan tahapan operasi ditulis dalam urutan alami menggunakan struktur kontrol standar..

Maka dapat disimpulkan bahwa integrasi pihak ketiga, khususnya melalui Telegram, menghasilkan layanan yang fleksibel, cepat, dan lebih mudah diterima pengguna.

a. Layanan *Bot* Telegram

Layanan *Bot* Telegram diimplementasikan sebagai akun otomatis yang dikendalikan program komputer untuk memproses dan merespons interaksi pengguna secara instan tanpa nomor telepon tambahan. *Bot* ini populer untuk penyebaran informasi otomatis karena mudah diintegrasikan dengan basis data kustom. (Sari, 2025:232)

menjelaskan bahwa *Bot* API memungkinkan pengembang membangun program yang berinteraksi lewat antarmuka percakapan, dengan penekanan pada kecepatan dan kemudahan akses data kustom. Fleksibilitas platform juga didukung ketersediaan *Bot Engine* berlisensi terbuka untuk integrasi dengan berbagai perangkat. Widya dan Airlangga (2020:14) menambahkan bahwa Telegram dipilih karena *Bot* API/*Bot* Engine-nya berlisensi terbuka dan mudah dihubungkan dengan berbagai teknologi. Meski demikian, keamanan kredensial *bot* harus dijaga ketat.

Dengan uraian tersebut, *Bot* Telegram menjadi instrumen otomatisasi yang andal untuk pelayanan informasi, asalkan keamanan token tetap terjaga.

13. Perangkat Pendukung Pemrosesan Citra

Kualitas ekstraksi fitur wajah oleh algoritma deep learning sangat ditentukan oleh kondisi citra yang diterima dari kamera. Video mentah kerap memiliki ukuran tidak seragam, distorsi warna, serta pencahayaan yang kurang ideal sehingga belum layak diproses langsung. Karena itu, pra-pemrosesan citra digital diperlukan untuk menstandarkan matriks visual sebelum klasifikasi (Febiyani et al., 2025:98). Manipulasi gambar dasar ini memanfaatkan pustaka computer vision tingkat rendah yang ringan. Pemrosesan pada perangkat meminimalkan beban komputasi dan mengurangi ketergantungan pada *server* terpusat (Abdullahu et al., 2025:1)

a. *Open Source Computer Vision Library* (OpenCV)

Open Source Computer Vision Library (OpenCV) merupakan pustaka gratis dan sumber terbuka untuk pengolahan citra, visi komputer, dan pembelajaran mesin. OpenCV mendukung beberapa bahasa pemrograman, termasuk Python, C++, dan Java, serta menyediakan modul untuk membaca dan menulis citra, menampilkan hasil melalui antarmuka grafis, melakukan transformasi dan manipulasi, menerapkan filter, serta menjalankan deteksi dan pengenalan wajah (Kumar et al., 2022:23-26). Dalam tahap prapemrosesan citra wajah, penyaringan digunakan untuk mengurangi derau dan memperbaiki kualitas citra sebelum proses deteksi. *Median filtering* bekerja dengan mengurutkan nilai piksel pada suatu lingkungan dan mengganti piksel pusat menggunakan nilai tengahnya. Pengujian pada citra wajah menunjukkan bahwa metode ini menghasilkan tingkat kesalahan yang lebih rendah dibandingkan *mean filtering* sehingga sesuai untuk mendukung kualitas masukan dalam proses deteksi wajah(Sunardi et al., 2023:292-295).

Dengan demikian, OpenCV menyediakan komponen pengolahan citra yang lengkap untuk mendukung pembangunan sistem visi komputer. Penerapan penyaringan pada tahap prapemrosesan membantu menghasilkan citra yang lebih bersih sehingga proses ekstraksi karakteristik dan pengenalan wajah dapat dilakukan menggunakan masukan visual yang lebih baik..

14. Arsitektur Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan arsitektur jaringan saraf tiruan dalam deep learning yang secara alami mengintegrasikan ekstraksi fitur visual dari tingkat rendah, menengah, hingga tingkat tinggi secara end-to-end dalam struktur multilapis. CNN terdiri atas lapisan konvolusi (convolution layer), lapisan pooling, fungsi aktivasi nonlinear, dan lapisan fully connected yang bekerja bersama untuk mengekstrak serta mengklasifikasikan pola pada citra masukan (Cong & Zhou, 2023:5). Arsitektur CNN bekerja dengan cara menggeser jendela berukuran tetap (kernel) pada seluruh area peta fitur (feature map) untuk mengekstrak pola spasial secara hierarkis. Melalui operasi konvolusi dan pooling yang berulang, CNN mampu mempelajari representasi fitur yang semakin abstrak dari lapisan ke lapisan, sehingga menjadi fondasi utama sistem pengenalan visual modern seperti deteksi wajah, pengenalan identitas, hingga *anti-spoofing* biometrik (Cong & Zhou, 2023:5–6).

a. Menakanisme Mekanisme Lokalisasi Objek pada YOLO26

Dalam arsitektur YOLO (You Only Look Once), CNN digunakan sebagai backbone untuk memprediksi bounding box dan kategori objek secara langsung dari seluruh citra dalam satu evaluasi jaringan. Setiap prediksi bounding box menghasilkan nilai confidence score yang mengukur seberapa yakin jaringan terhadap

keberadaan objek sekaligus akurasi lokasi prediksi (Cong & Zhou, 2023:22). Nilai kepercayaan dihitung menggunakan rumus berikut:

Persamaan 2. 1 Confidence Score YOLO untuk Deteksi Objek

$$\text{Confidence Score} = Pr(\text{Object}) \times IoU_{pred}^{truth} \times Pr(\text{Class}_i \mid \text{Object}) \qquad 2.1$$

Keterangan:

Pr(Object) = probabilitas bahwa suatu objek (wajah) benar-benar ada di dalam bounding box

IoU_t^{ruth} / IoU_p^{red} = Intersection over Union antara *bounding box* prediksi dan *ground truth*.

$Pr(Class_i \mid Object)$ = probabilitas kondisional bahwa objek termasuk kelas ke-i, diberikan objek ada

Confidence Score = nilai kepercayaan akhir gabungan keberadaan dan akurasi lokasi objek

Sumber: Cong & Zhou (2023:22).

Komponen *Pr(Object)* merepresentasikan kemungkinan keberadaan objek pada *bounding box*. Nilai *IoU* mengukur rasio tumpang-tindih prediksi dengan *ground truth*, sedangkan $Pr(Class_i|Object)$ menyatakan probabilitas kelas kondisional. Perkalian ketiga komponen menghasilkan skor kepercayaan akhir yang digunakan untuk menentukan apakah suatu region mengandung wajah beserta lokasi presisinya (Cong & Zhou, 2023:22).

b. Mekanisme Pembentukan Face Embedding pada ArcFace

ArcFace menggunakan *Deep Convolutional Neural Network* (DCNN) berbasis ResNet sebagai backbone untuk memetakan citra wajah ke dalam ruang *embedding* 512 dimensi. ArcFace menambahkan penalti margin angular aditif m pada sudut antara fitur citra dan bobot kelas ground truth untuk memperketat batas antarkelas pada ruang hipersfer, sehingga meningkatkan daya diskriminasi model pengenalan wajah Deng et al.,(2019:4688). Fungsi loss ArcFace dirumuskan sebagai berikut:

Persamaan 2. 2 *Additive Angular Margin Loss* (ArcFace)

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\frac{e^{s\cdot\cos(\theta_{y_i}+m)}}{e^{s\cdot\cos(\theta_{y_i}+m)}+\sum_{j=1,\,j\neq y_i}^{n}e^{s\cdot\cos\theta_j}}\right] \qquad 2.2$$

Keterangan:

L = nilai fungsi kerugian ArcFace yang diminimalkan selama pelatihan

N = jumlah total sampel dalam satu batch pelatihan

s = skala fitur (feature scale), umumnya diset 64

θyi = sudut antara vektor fitur citra ke-i dan bo*bot* kelas ground truth yi

m = penalti margin angular aditif, umumnya diset 0,5 radian

n = jumlah total kelas identitas dalam *dataset* pelatihan

θj = sudut antara vektor fitur citra dengan bo*bot* kelas ke-j

Sumber: Deng et al. (2019:4688).

Persamaan ArcFace menunjukkan bahwa margin m ditambahkan langsung pada sudut θyi sebelum fungsi kosinus dihitung, sehingga mendorong jaringan untuk meminimalkan jarak intra-kelas sekaligus memaksimalkan jarak antarkelas. Skala s berfungsi meregulasi distribusi logit agar gradient tetap informatif selama pelatihan (Deng et al., 2019:4688)

c. Mekanisme Deteksi *Face Anti-Spoofing* pada MiniFASNet

MiniFASNet merupakan arsitektur CNN ringan. Prinsip *depthwise separable convolution* memisahkan konvolusi menjadi *depthwise convolution* dan *pointwise convolution* untuk mengurangi ukuran model dan beban komputas (Cong & Zhou, 2023:17). Tahap akhir jaringan menggunakan lapisan fully connected (FC) untuk mengklasifikasikan fitur yang diekstrak menjadi dua kategori: wajah asli (bona fide) atau wajah palsu (*spoof*ing). Operasi matematis lapisan FC yang menjadi inti klasifikasi biner ini dirumuskan sebagai berikut (Cong & Zhou, 2023:7):

Persamaan 2. 3 *Fully Connected Layer* Klasifikasi *Biner* MiniFASNet

$$u_j^l = \max\left(0,\ \sum_i y_i^l \cdot w_{ij}^l + b_j^l\right) \qquad 2.3$$

Keterangan:

u_j^l = aktivasi bersih neuron ke-j pada lapisan FC ke-l

y_i^{l-1}= nilai keluaran dari lapisan depthwise separable convolution sebelumnya

w_{ij}^{l}= bo*bot* koneksi dari neuron ke-i menuju neuron ke-j pada lapisan FC ke-l

b_j^l = bias pada lapisan fully connected ke-l

max(0, ·) =fungsi aktivasi ReLU yang menekan nilai negatif menjadi nol

l = indeks lapisan dalam arsitektur CNN MiniFASNet

Sumber: Cong & Zhou (2023:7).

Persamaan di atas menggambarkan mekanisme lapisan fully connected yang digunakan MiniFASNet untuk menghasilkan keputusan klasifikasi biner. Nilai u_j^l merupakan kombinasi linear dari fitur masukan y_i^l yang telah diekstrak oleh lapisan-lapisan depthwise separable convolution sebelumnya, dikalikan bo*bot* w_{ij}^l dan ditambah bias b_j^l. Fungsi ReLU max(0,·) memastikan hanya aktivasi positif yang diteruskan untuk menentukan apakah wajah termasuk kategori asli atau serangan *spoof*ing (Cong & Zhou, 2023:7).

B. Kajian Empiris

Pengembangan *Dosen Tracker* berbasis pengenalan wajah menuntut arsitektur yang akurat secara biometrik, responsif dalam memperbarui lokasi dosen, sekaligus aman terhadap manipulasi visual (*presentation attack*). Untuk menegaskan posisi dan kebaruan penelitian ini, dilakukan telaah kritis terhadap sejumlah studi terdahulu, mulai dari sisi model deteksi, infrastruktur peladen, hingga keandalan kode..

Pada sisi model, Susanti et al. (2023, dalam Putriyani , 2025:326) menerapkan YOLOv5 untuk mendeteksi banyak wajah mahasiswa secara *real-time*, sedangkan keterbatasan deteksi pada jarak, pencahayaan gelap, dan sudut wajah masih menjadi tantangan sistem pengenalan wajah. Berangkat dari keterbatasan itu, Putriyani, (2025:325) mengintegrasikan YOLOv8 dengan ArcFace yang dilengkapi *anti-spoofing* untuk mendeteksi dan mengidentifikasi wajah mahasiswa secara otomatis dan *real-time*, sementara Chakrabarty (2026:7) mengulas YOLO26 berarsitektur *NMS-free end-to-end* yang dilaporkan mempercepat inferensi pada CPU sekitar 43% dibandingkan *baseline* berbasis NMS. Aspek lokasi juga tidak diabaikan: Susatyono et al. (2026:242) mengombinasikan *geolocation* untuk memvalidasi lokasi dengan pengenalan wajah untuk memverifikasi identitas pengguna. Meski demikian, studi-studi presensi tersebut masih berorientasi pada pencatatan kehadiran; celah inilah yang diarahkan penelitian ini pada pemetaan *last seen location* dosen dan distribusinya kepada mahasiswa.

Tidak hanya pada sisi model, tantangan juga muncul di tingkat peladen yang menentukan efisiensi sumber daya. Uji beban Santiago dan Benisius (2025:1225) dengan Apache JMeter menunjukkan Gin paling efisien (memori 26,74 MB; CPU 0,03%), disusul Express.js (67,53 MB; 0,03%), sedangkan FastAPI menggunakan sumber daya tertinggi (79,90 MB; 0,04%) dan mencatat *throughput* lebih rendah pada mayoritas skenario. Meskipun demikian, Suryotomo et al. (2024:120) menunjukkan

bahwa untuk inferensi *machine learning* yang bersifat *compute-heavy*, penambahan *core* prosesor dan *preemptive model loading* dapat meningkatkan *throughput*, sementara FastAPI mempertahankan tingkat eror di bawah 20% pada beban yang melampaui proyeksi awal 250 pengguna.

Selain performa, mutu perangkat lunak juga bergantung pada keandalan kode. Penerapan *Test-Driven Development* melalui siklus *Red-Green-Refactor* oleh Agustin et al. (2026:11) pada arsitektur *microservices* meningkatkan *statement coverage* menjadi 96,81% dan menurunkan estimasi *defect rate* menjadi 4,33 per 1.000 baris kode, sedangkan bagi pengembang tunggal, metode *Personal Extreme Programming* (PXP) yang dikaji Sidiq et al. (2025:37) menekankan komunikasi dengan klien dan fleksibilitas terhadap perubahan

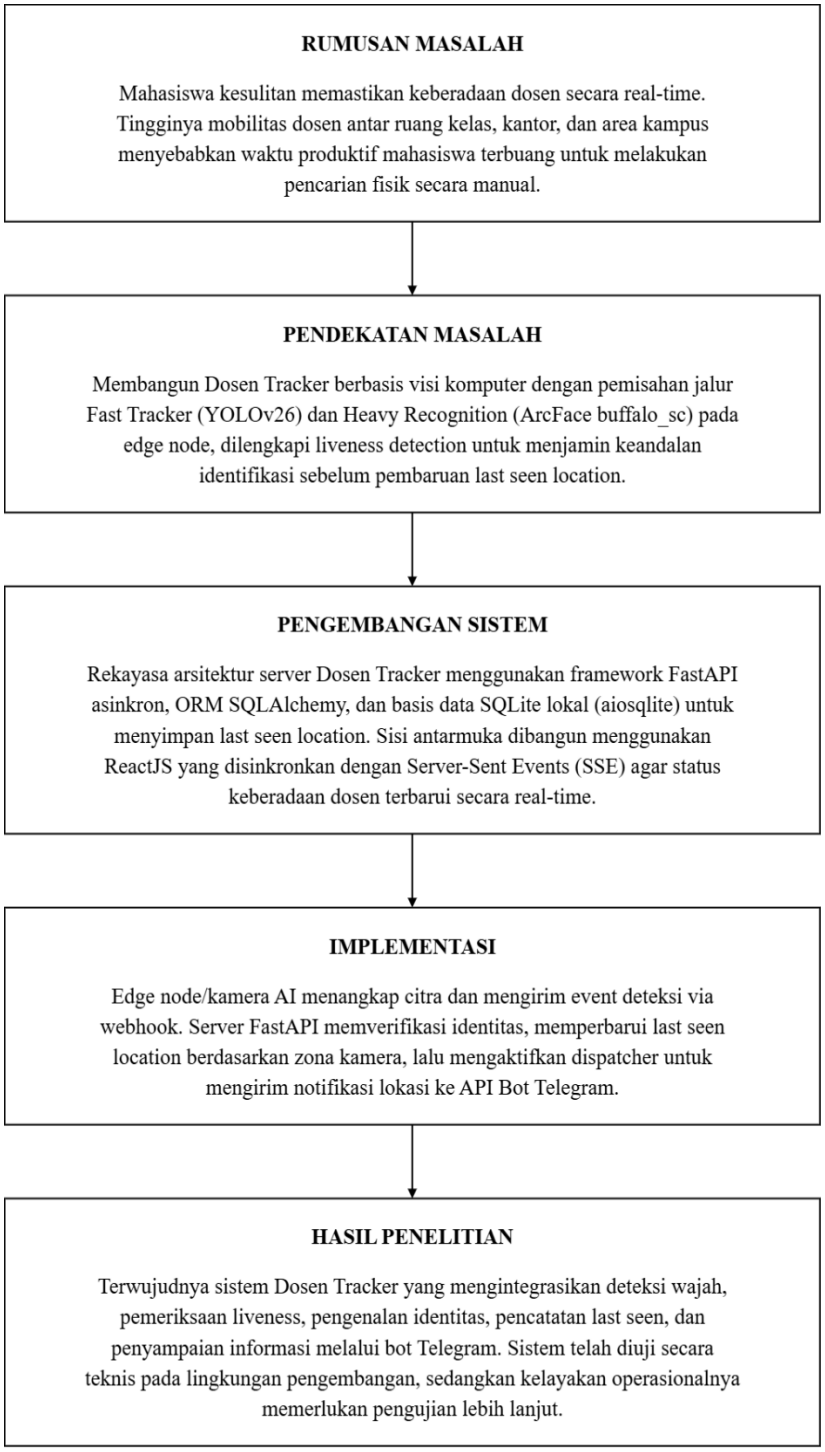
Berpijak pada seluruh temuan tersebut, penelitian ini menempatkan pengenalan wajah sebagai fondasi pelacakan keberadaan dosen secara aktif, bukan sebagai instrumen absensi formal. Melalui *liveness detection*, sistem memvalidasi identitas sebelum memperbarui *last seen location* berbasis zona kamera ke basis data SQLite melalui ORM SQLAlchemy asinkron, lalu mendistribusikan perubahannya kepada mahasiswa secara *real-time* melalui *event-driven webhook* pada *bot* Telegram. Pendekatan *cross-platform* inilah yang membedakan penelitian ini dari riset biometrik konvensional: tujuannya bukan sekadar mengejar akurasi matematis, melainkan menyelesaikan masalah nyata inefisiensi

waktu mahasiswa dalam melacak keberadaan dosen di lingkungan kampus.

C. Kerangka Berfikir

Kerangka berpikir penelitian ini disusun untuk menunjukkan alur penyelesaian masalah secara sistematis, mulai dari identifikasi persoalan hingga hasil yang diharapkan. Permasalahan utama berangkat dari kesulitan mahasiswa dalam mengetahui keberadaan dosen secara *real-time* akibat tingginya mobilitas dosen di lingkungan kampus. Masalah tersebut direspons melalui pembangunan *Dosen Tracker* berbasis visi komputer yang memisahkan jalur *Fast Tracker* (YOLOv26) dan *Heavy Recognition* (ArcFace), serta didukung *liveness detection*..

Pada tahap pengembangan, sistem dirancang menggunakan arsitektur *server* asinkron berbasis FastAPI, ORM SQLAlchemy, dan basis data SQLite, dengan antarmuka ReactJS yang disinkronkan melalui *Server-Sent Events* (SSE). Tahap implementasi kemudian mencakup integrasi *edge node*/kamera AI, verifikasi identitas, pembaruan *last seen location*, dan distribusi notifikasi melalui API *Bot* Telegram. Hasil akhirnya adalah terwujudnya ekosistem *Dosen Tracker* yang andal, tahan terhadap *spoofing*, dan mampu menyampaikan informasi lokasi terakhir dosen kepada mahasiswa secara efisien. Dengan demikian, kerangka berpikir ini menunjukkan hubungan yang logis antara masalah, solusi, proses pengembangan, dan capaian penelitian. Alurnya dapat dilihat pada Gambar 2.10



Gambar 2. 10 Kerangka Berfikir