BAB II

KAJIAN TEORITIS

A. Kajian Teoritis

1. Algoritma *YOLOv8* (*You Only Look Once*)

   Metode *YOLOv8* adalah algoritma *State – of – the – Art* yang digunakan untuk mengevaluasi dan mendeteksi objek secara *real – time.* Untuk menunjukkan besarnya tidak prediksi, YOLO memformulasikan deteksi objek sebagai satu masalah regresi tunggal (*Singgle neural network*). Nilai kepastian prediksi ini disebut dengan *Confidence Score*. Nilai tetinggi adalah 1,0 yang berarti sangat presisi, dan terendah adalah 0 yang berarti tidak terdekteksi.

   Nilai *Confidence Score* di formulasikan menggunakan persamaan matematika yang memperhitungkan probabilitas dan tingkat tumpeng tindih kontak prediksi (Agustien et al., 2021).

$$Confidence = Pr(Object)\ x\ \text{IoU}_{pred}^{truth}$$

   Keterangan :

   Pr (*Object*) = Probabilitas keberadaan objek dalam kontak pembatasan (*bounding box*).

   $\text{IoU}_{pred}^{truth}$ = *Intersection over Union*, yaitu persentase tumpang tindih antara *bounding box* prediksi dengan data kebenaran actual (*ground truth*).

Menurut penelitian yang di lakukan (Isa Mahfudi et al., 2024), pengaturan nilai ambang batas pada *confidence score* memiliki pengaruh yang sangat signifikan terhadap akurasi akhir deteksi YOLO. Jika nilai di tentukan terlalu tinggi, sistem akan menjadi sengat selektif sehingga berisiko melewatkan objek yang sederhana ada (*false negative*). Sebaliknya, jika terlalu rendah, sistem akan memicu munculnya banyak deteksi palsu (*false positive*). Oleh sebab itu penentuan *confidence score* yang seimbang sangat krusial agar sistem dapat bekerja secara optimal dalam mengenali objek target.

Sementara itu, penelitian yang di lakukan oleh (Taufiqurrahman et al., 2024) menjelaskan bahwa nilai ambang batas dari *confidence score* memiliki pengaruh yang sangat vital terhadap stabilitas performa deteksi di lingkungan nyata. Pengaturan parameter yang seimbang pada arsitektur YOLO sangat krusial guna menyaring kotak – kotak pembatasan yang lemah dan mempertahankan akurasi lokalisasi objek yang tinggi, terutama saat sistem di hadapkan pada tantangan variasi kondisi lingkungan sekitar objek yang dinamis.

2. *Deep Learning*

Deep Learning atau pembelajaran mendalam merupakan bagian dari kecerdasan buatan dan *machine learning* yang merupakan pengembangan dari *neural network multiple layer* untuk memberikan ketepatan tugas seperti deteksi objek, pengenalan suara, terjemah bahasa dan lain – lain. *Deep Learning* berbeda dari teknik *machine learning*

yang tradisonal, karena *deep learning* secara otomatis melakukan representasi dari data seperti gambar, video, atau text tanpa memperkenalkan aturan kode atau pengetahuan domain manusia (Raup et al., 2023).

Dalam penelitian lain yang di lakukan oleh (Nur Akmal & Maelasari, 2025) *Deep Learning* digunakan untuk melacak identifikasi dan perilaku siswa secara otomatis melalui penggunaan algoritma canggih, seperti *Convolutional Neural Networks* (CNN). Sistem ini mengindentifikasi mood siswa (seperti marah, senang atau takut) dan memonitor tingkat keterlibatan mereka selama kelas daring.

Selain di bidang pendidikan, keunggulan *Deep Learning* juga sangat krusial dalam sistem keamanan visual dan deteksi wajah. Menurut penelitian yang di lakukan oleh (Yanto et al., 2023) , arsitektur *Deep Learning* seperti model algoritma YOLO (*You Only Look Once*) terukti sangat tangguh dalam mendeteksi dan mengenali wajah, bahkan ketika objek berada dalam kondisi tertutup seperti saat menggunakan masker. Model YOLO mampu mengekstraksi fitur wajah dengan tingkat komputasi yang cepat dan akurasi yang tinggi secara *real-time*.

Berdasarkan hasil penelitian tersebut, dapat di simpulkan bahwa metode *deep learning* merupakan pendekatan mutakhir dalam domain kecerdasan buatan dan *machine learning* yang memanfaatkan jaringan saraf tiruan multikapasitas (*multiple hidden layers*) untuk memproses serta merepresentasikan data visual secara otomatis. Proses ini tidak

memerlukan pemrograman aturan kode secara manual, melainkan sistem secara mandiri belajar langsung dari karakteristik data yang diberikan. Metode ini terbukti sangat adaptif dan andal untuk diterapkan pada berbagai bidang komputasi visual yang kompleks, mulai dari analisis citra dinamis hingga deteksi objek pada kondisi wajah yang tertutup sebagian (*occlusion*). Dalam penelitian ini, keunggulan *deep learning* melalui algoritma YOLOv8 dimanfaatkan untuk mengotomatisasi sistem deteksi dini kepatuhan penggunaan Alat Pelindung Diri (APD) pekerja secara *real-time* dan sistematis di Hanggar Perakitan PT Industri Kereta Api (PT INKA).

2. Kondisi Wajah Tertutup Sebagian (*Occlusion*)

Pada dasarnya, *occlusion* adalah kondisi di mana sebagian objek yang ingin di deteksi terhalang oleh objek lain yang menutupi fitur utamanya. Dalam konteks sistem keamanan, occlusion sering disebabkan oleh penggunaan atribut pelindung seperti masker, kacamata, atau helm, yang mengakibatkan tertutupnya informasi spasial penting dari wajah sehingga menyulitkan proses identifikasi pada komputer (Ibnu Halim Mustofa & Edy Winarno, 2023).

Menurut (Hardiansyah & Primasetya, 2023), meskipun teknologi deteksi wajah sudah berkembang pesat, performa akurasi sistem konvensional sering kali menurun dratis ketika di hadapkan pada kondisi wajah bermasker (occlusion). Hal ini diakibatkan hilangnya fitur ensial dari wajah.

Hal senada juga di ungkapkan dalam penelitian oleh (Ahsan Saputra et al., 2025) yang menyatakan bahwa tantangan utama pada wajah yang mengalami *occlusion* adalah hilangnya informasi geometri penting seperti area hidung, mulut dan rahang. Oleh karena itu, di perlukan sistem yang mampu focus melakukan ekstraksi fitur pada area penanda wajah (*facial landmarks*) yang tidak tertutup, seperti area mata, alis dan dahi agar klasifikasi identitas tetap dapat di kenali secara presisi

Kesimpulan dari pernyataan tersebut menunjukkan bahwa *occlusion* adalah suatu kondisi teknis yang menantang dalam pengolahan citra digital. Algoritma tradisional sangat rentan mengalami kegagalan deteksi, sehingga membutuhkan pendekatan *Deep Learning* tingkat lanjut yang telah terbukti tangguh dalam mengekstraksi dan memetakan sisa area wajah yang masih terlihat secara akurat.

3. Keselamatan dan Kesehatan Kerja (K3) & Alat Pelindung Diri (APD)

Sistem Keselamatan dan Kesehatan Kerja (K3) memegang peran yang sangat vital dalam mereduksi risiko operasional di lingkungan industri manufaktur berat. Perlindungan ini tidak hanya berorientasi pada pemenuhan regulasi hukum, melainkan juga berfokus pada jaminan keselamatan jasmani dan rohani tenaga kerja agar terhindar dari kecelakaan kerja yang fatal serta penyakit akibat paparan aktivitas produksi (Dinda Ayuni Cantika & Mohammad Sofyan, 2024)

Kebijakan penegakan keselamatan di area kerja yang berisiko tinggi umumnya mengandalkan Alat Pelindung Diri (APD) sebagai benteng pertahanan terakhir (*last line of defense*). Atribut perlindungan wajah dan kepala termasuk helm proyek, kacamata keselamatan (*safety glasses*), dan masker pelindung pernapasan wajib dikenakan secara konsisten oleh seluruh pekerja untuk mengantisipasi potensi bahaya benturan fisik, paparan radiasi las, hingga masuknya partikel debu mikro ke saluran pernapasan (Rakhmawati et al., 2023).

Kendala utama yang sering dihadapi di area hanggar perakitan yang luas adalah rendahnya efisiensi pengawasan kepatuhan APD jika hanya dilakukan secara manual. Keterbatasan jumlah personel pengawas keselamatan (*Safety Officer*) di lapangan membuat pemantauan terus-menerus terhadap ratusan pekerja menjadi sulit direalisasikan, sehingga membuka peluang terjadinya kelalaian yang membahayakan jiwa pekerja (Amilah Eka Putri et al., 2025).

Guna mengatasi celah pengawasan tersebut, integrasi teknologi *computer vision* berbasis model *deep learning* YOLOv8 hadir sebagai solusi pemantauan otomatis. Sistem cerdas ini memanfaatkan fenomena oklusi visual (wajah tertutup APD) bukan sebagai penghambat, melainkan sebagai parameter deteksi utama untuk mengidentifikasi kepatuhan pekerja secara otomatis dan memberikan peringatan dini (*early warning*) secara *real-time*.

4. Website

Website merupakan fasilitas internet yang menghubungkan dokumen dalam lingkup lokal maupun jarak jauh. Dokumen pada website disebut dengan *webpage* dan link dalam *website* memungkinkan pengguna bisa berpindah dari satu *page ke page* lain (*hypertext*), baik diantara page yang disimpan dalam server yang sama maupun server diseluruh dunia. Pages diakses dan dibaca melalui *browser* seperti *Netscape Navigator, Internet Explorer, Mozila Firefox*, Google Chrome dan aplikasi browser lainnya (Zulfa & Arifudin, 2025).

a. *Framework*

Framework adalah kerangka kerja yang menyediakan struktur dasar untuk pengembangan perangkat lunak. Dalam konteks CodeIgniter, framework kini dirancang untuk mempermudah pembuatan aplikasi web berbasis PHP dengan prinsip-prinsip yang mempromosikan pengkodean yang bersih terorganisasi dan dapat dipelihara (Syelly, 2025).

Framework merupakan wadah atau kerangka kerja utama dari sebuah *website* yang akan dibangun. Berdasarkan penelitian dari (Rizki Akbar et al., 2024), penerapan *framework* dalam pembuatan aplikasi berbasis web secara nyata mampu meningkatkan efisiensi waktu pengembangan dan menyederhanakan kompleksitas pengkodean. Hal tersebut dikarenakan *framework* menyediakan komponen serta arsitektur yang terstruktur, sehingga ketika sistem

memerlukan pemeliharaan atau perbaikan di masa mendatang, proses pelacakan dan pembetulan baris kode (*debugging*) dapat dilakukan secara lebih mudah dan cepat.

Framework berfungsi sebagai wadah atau cetakan dasar utama yang dirancang untuk mempercepat seluruh proses pembangunan aplikasi *website*. Menurut penelitian dari (Nurdianto & Sutaji, 2025), implementasi *framework* CodeIgniter secara nyata terbukti meningkatkan efisiensi pengembangan sistem dengan waktu penyelesaian 30% lebih cepat dibandingkan metode pemrograman *native*. Struktur arsitektur yang terorganisasi dengan baik pada *framework* ini juga memberikan keuntungan besar saat sistem memerlukan pemeliharaan atau perbaikan, sehingga pengembang dapat dengan mudah menemukan serta memperbaiki letak kesalahan secara praktis tanpa harus menghabiskan banyak waktu untuk mengonfigurasi ulang baris kode dari awal.

Berdasarkan pendapat diatas, dapat di simpulkan bahwa *framework* adalah kerangka kerja dasar yang dirancang untuk mempermudah dan mempercepat waktu pengembangan perangkat lunak, khususnya berbasis website. Dengan menyedikan wadah dan struktur pengkodean yang terorganisasi, *framework* membantu pengembangan menghasilkan penulisan kode yang bersih, rapi, dan mudah di pelihara. Penggunaan kerangka kerja ini juga dinilai

sangat efisien karena dapat menyederhanakan proses perbaikan serta pemeliharaan sistem di masa mendatang.

b. *React Js*

ReactJs adalah sebuah pustaka (*library*) JavaScript yang di kembangkan oleh facebook untuk membangun antarmuka pengguna (*User Interface /* UI). Pendekatan utama dalam *ReactJs* adalah konsep arsitektur berbasis komponen, dimana sebuah tampilan *website* atau aplikasi di pisah menjadi beberapa bagian kecil yang berdiri sendiri. Menurut (Kevin Juan & Budi, 2023), *React JS* menggunakan konsep komponen yang akan selalu menghasilkan sebuah elemen HTML untuk di tampilkan di *Browser*. Fungsionalitas ini memudahkan pengembangan untuk menggunakan ulang komponen tersebut (*reusable*), sehingga pembuatan tampilan menu antarmuka yang responsif dan efisien dapat di lakukan dengan lebih cepat.

Pendapat lain di kemukakan oleh (Makhfuddin Akbar et al., 2026), yang menjelaskan bahwa *ReactJS* sangat ideal untuk membangun aplikasi modern berbasis *Single Page Application* (SPA). *ReactJS* menggunakan teknologi *Virtual Document Object Model* (*Virtual DOM*) yang mampu meningkatkan responsivitas aplikasi secara signifikan. Ketika terdapat perubahan data, sistem tidak perlu memuat ulang (*reload*) keseluruhan halaman *web,*

melainkan hanya merender komponen – komponen spesifik yang mengalami perubahan saja.

Hal ini didukung oleh penelitian dari (Miftah Faroq Santoso, 2021), yang menunjukkan bahwa dengan mengadopsi teknik SPA berbasis ReactJS, *browser* tidak akan melakukan *load* ulang secara terus-menerus saat bernavigasi karena data diolah secara otomatis oleh pustaka tersebut, sehingga menghasilkan performa pemuatan halaman yang jauh lebih cepat serta memberikan pengalaman interaksi yang lebih nyaman bagi pengguna.

Bedasarkan pendapat di atas, dapat di simpulkan bahwa *ReactJS* adalah pustaka JavaScript berbasis komponen yang di rancang untuk mempercetpat dan mempermudah proses pengembangan antarmuka pengguna pada aplikasi *web.* Dengan memanfaatkan fitur *Virtual Dom, ReactJS* memberikan efisiensi tinggi dalam pemrosesan data layar sehingga performa *website* menjadi jauh lebih ringan, interaktif, dan mulus tanpa harus memuat ulang halaman secara keseluruhan.

c. Python

Python adalah salah satu bahasa pemrograman yang biasa menggunakan *interpreter* dalam kode programnya. *Interpreter* dapat menerjemahkan kode secara langsung, dan Python dapat dijalankan di berbagai platform, seperti Windows, Linux dan lainnya (Rizki et al., 2025).

Pendapat lain menurut (Pannadhitthana Candra, 2025), Python adalah bahasa pemrograman tingkat tinggi yang sangat populer, di kenal karena sintaksnya yang sederhana dan mudah di pahami. Dikenalkan pertama kali oleh Guido van Rossum pada tahun 1991, Python telah paling banyak di gunakan di dunia, terutama dalam bidang data science, pengembangan web dan otomatisai.

Hal ini sejalan dengan penelitian dari (Slamet & Anistyasari, 2021), yang menegaskan bahwa popularitas Python sebagai salah satu bahasa pemrograman yang paling diminati didorong oleh fleksibilitasnya yang tinggi. Python menyediakan lingkungan pengembangan yang sangat ramah bagi pengguna karena kesederhanaan sintaksnya, sehingga mampu mempercepat efisiensi waktu dalam penulisan baris kode sistem.

Dari pendapat di atas, dapat di simpulkan bahwa Python adalah Bahasa pemrograman tingkat tinggi yang sangat populer karena memiliki sintaks yang sederhana, mudah di pahami, serta bersifat lintas platform (*cross – platfrom*) sehingga dapat di jalankan di berbagai sistem operasi Windows dan *Linx*. Sebagai bahasa yang menggunakan sistem *interpreter* untuk menerjemahkan kode secara langsung, Python menawarkan efisiensi tinggi dalam pengembangan perangkat lunak, terutama pada bidang – bidang strategis seperti data *science*, pengembangan web, dan sistem otomatisasi. Kesederhanaan dan fleksibilitasnya

menjadikan Python sebagai bahasa yang paling banyak digunakan baik oleh pemula maupun profesional untuk memecahkan berbagai masalah komputasi modern.

d. Laragon

Laragon merupakan solusi ideal bagi pengembangan yang membutuhkan lingkungan server local yang ringan dan fleksibel. Menurut (Rambe et al., 2025) Laragon adalah perangkat lunak yang bersifat *open source* (terbuka) yang dapat mendukung banyak sekali sistem operasi dimana Laragon bertugas sebagai *server virtual* atau sering di sebut sebagai *localhost*.

Laragon adalah solusi pengembangan web yang menyediakan lingkungan *server local* yang lengkap, ringan, fleksibel, dan portabel, sehingga memudahkan pengembangan dalam membangun dan mengelola aplikasi berbasis web (Ni'mah & Bhakti, 2025) .

Hal ini sejalan dengan hasil analisis dari (Satriyo Rizkyansah & Muhammad Tri Putro Alfaz Dzikri, 2022), yang menunjukkan bahwa Laragon sangat andal digunakan sebagai *local server development tool* karena memiliki keunggulan performa yang stabil serta manajemen *resource* memori yang jauh lebih efisien untuk proses eksekusi kode *website*.

Dari pendapat di atas, dapat di simpulkan bahawa Laragon adalah perangkat lunak *open – source* yang berfungsi sebagai

penyedia lingkungan server local (*localhost*) yang lengkap, ringan, dan portabel. Sebagai server virtual, laragon memberikan fleksibilitas tinggi bagi pengembangan karena mendukung berbagai sistem operasi dan mempermudah proses pengelolaan serta pembangunan aplikasi berbasis web tanpa membebani sumber daya perangkat secara berlebihan. Fleksibilitas dan kemudahan instalasinya menjadikannya solusi ideal untuk menciptakan alur kerja pengembangan web yang lebih efisien dan terorganisasi.

5. Database

Database dapat diibaratkan sebagai wadah penyimpanan terstruktur yang berfungsi untuk mengumpulkan dan menyimpan berbagai informasi secara sistematis agar mudah diakses. Data-data tersebut diatur di dalam media penyimpanan komputer sedemikian rupa sehingga memungkinkan pengolahan informasi yang efisien, aman, dan terorganisasi. Dengan adanya basis data, sebuah sistem tidak sekadar menjadi tempat penyimpanan elektronik biasa, melainkan mampu menjamin ketersediaan data secara cepat serta mencegah pengulangan data (*redundansi*) (Efendi et al., 2023).

Tanpa perlu melakukan pengolahan data secara manual, pengguna kini dapat memanfaatkan sistem manajemen terpadu untuk mengelola basis data tersebut. Sistem ini dilengkapi dengan fitur yang menjamin keamanan, konsistensi, keutuhan, dan perlindungan data dari akses

yang tidak sah. Untuk mengendalikan pembuatan, pemeliharaan, dan pemanfaatan data dalam skala besar, digunakanlah perangkat lunak yang disebut *Database Management System* (DBMS). Istilah DBMS ini merujuk pada sistem yang dibuat secara khusus untuk menyusun dan mengelola kumpulan informasi yang saling berkaitan guna memenuhi berbagai kebutuhan operasional suatu organisasi, sebagaimana dijelaskan oleh (Hadie Ahsa et al., 2023).

Lebih lanjut, impelementasi *Database Management System* (DBMS) juga memiliki peranan yang sangat vital dalam dalam mengakomodasi kompleksitas data agar mudah di proses secara terstruktur. Berdasarkan kajian dari (Annisa Rahmawita et al., 2023), fungsi, utama dari perangkat lunak DBMS tidak hanya sebatas sebagai pusat penyimpanan (*data storage*), melainkan juga untuk mengelola berbagai jenis data yang bervariasi dan kompleks. Penggunaan sistem manajemen ini dinilai sangat krusial untuk menjembatani pengguna akhir dengan basis data secara efesien, sehingga seluruh informasi dapat di proses, dievaluasi, dan di jaga konsistensi serta keamanannya agar sistem mampu bersaing di era digital.

6. *Mysql*

Mysql merupakan sumber perangkat lunak manajemen *database* terbuka untuk menambahkan, memperbarui, menghapus dan menampilkan data. *Mysql* diklasifikasi sebagai bahasa SQL (*Structure Query Language*) yang memiliki beberapa perintah yang umum di

gunakan, yaitu memiliki beberapa perintah yang umum di gunakan, yaitu pilih *Insert, update* dan menghapus (Nabila et al., 2025).

MySQL adalah sistem manajemen basis data (DBMS) berbasis *open source* yang menggunakan *Structured Query Language* (SQL) sebagai bahasa utama untuk mengelola dan mengakses data. *MySQL* dikenal karena kemudahan penggunaan, kecepatan kinerja query, dan kecukupan untuk kebutuhan skala menengah hingga kecil. Selain itu, *MySQL* mendukung aplikasi multi-pengguna berbasis jaringan, menjadikannya populer untuk pengelolaan *database* pada situs web dan aplikasi berbasis internet. Sifat *open source* MySQL memungkinkan pengguna untuk mendistribusikan, menggunakan, dan memodifikasi perangkat lunak ini tanpa biaya, baik untuk keperluan pribadi maupun komersial. Dengan kemampuannya yang fleksibel dan kompatibel dengan standar *SQL, MySQL* menjadi salah satu pilihan utama dalam pengelolaan *database* (Rusyadi et al., 2024).

Implementasi MySQL tidak hanya sebatas pada manajemen data berskala kecil, melainkan juga sangat krusial dalam mendukung keandalan sistem aplikasi yang lebih kompleks. Berdasarkan penelitian dari (Suhatman et al., 2024), untuk meningkatkan proses kerja agar berjalan efektif dan efisien, sangat di perlukan kinerja sistem basis data MySQL yang optimal serta tangguh terhadap risiko pemadaman layanan (*downtime*). Kemudahan konfigurasi pada MySQL membuktikan bahwa DBMS ini mamptu diintegrasikan dengan

berbagai metode pengelolaan tingkat lanjut untuk memastikan layanan basis data tetap stabil , terpusat , dan terlindungi dengan baik.

7. *Flowchart*

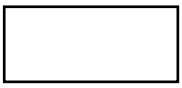
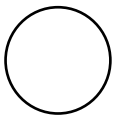
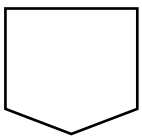
Menurut (Septiansyah et al., 2024) *Flowchart* atau bagan alur adalah diagram yang menampilkan langkah – langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah di gambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah.

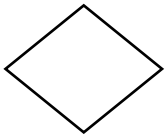
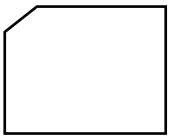
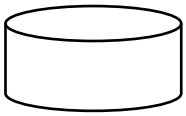
Flowchart adalah gambaran berupa grafik yang memiliki urutan suatu proses atau langkah-langkah secara sistematis untuk menjalankan suatu program. Flowchart dapat memberi gambaran untuk dilakukan proses analisis, perancangan, pengkodean dalam memecahkan masalah yang lebih terperinci dalam proses operasional suatu kegiatan. *Flowchart* umumnya digunakan untuk memudahkan penyelesaian masalah saat dilakukan evaluasi lebih lanjut. Dalam pengertian lain, *flowchart* dapat dijelaskan sebagai suatu diagram yang menggunakan simbol-simbol grafis untuk menggambarkan alur dari suatu proses. Diagram ini menunjukkan beberapa langkah atau tindakan yang direpresentasikan oleh simbol-simbol tersebut (Kus Indrani Listyoningrum et al., 2023).

Penggunaan bagan alur ini sangat penting untuk memetakan interaksi instruksi di dalam suatu perancangan sistem informasi. Menurut (Triana et al., 2024) *flowchart* pada dasarnya adalah suatu

bagan dengan simbol-simbol khusus yang berfungsi untuk menggambarkan urutan jalannya proses secara mendetail, sekaligus memperlihatkan hubungan yang jelas antara satu instruksi (proses) dengan proses lainnya di dalam sebuah program. Melalui pemetaan visual ini, alur logika program dapat dipahami dengan mudah sehingga meminimalisir kesalahan interpretasi selama tahap pengembangan sistem.

Tabel 2. 1 Simbol Flowchart

SIMBOL	NAMA	KETERANGAN
	*Input/Output*	Menunjukkan proses pemasukan data ke sistem atau hasil keluaran berupa informasi dari proses tersebut.
	Proses	Menunjukkan langkah atau operasi yang dilakukan dalam pengolahan data di suatu sistem.
	*Connector*	Menghubungkan alur proses yang terputus dalam halaman flowchart yang sama, agar alur tetap jelas dan teratur.
	*Off-page Connector*	Menghubungkan alur proses yang berpindah ke halaman flowchart lain, sehingga keterkaitan antarbagian tetap mudah diikuti.

	Anak Panah	Menunjukkan arah atau urutan aliran proses dalam flowchart.
	Keputusan	Menunjukkan titik pengambilan keputusan dengan dua atau lebih kemungkinan hasil (ya/tidak).
	*Preparation*	Pemberian harga awal
	*Terminal Point*	Menunjukkan awal atau akhir dari suatu flowchart.
	*Punched Card*	Melambangkan proses input atau output menggunakan kartu berlubang (metode lama).
	Dokumen	Menunjukkan data keluaran dalam bentuk dokumen cetak.
	Manual *Input*	Melambangkan data yang dimasukkan secara manual melalui keyboard.
	*Database*	Menunjukkan proses penyimpanan atau pengambilan data dari basis data.

Sumber : (Vira Agustin Pratiwi et al., 2025)

8. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa standar yang digunakan untuk memodelkan perangkat lunak. UML dapat digunakan untuk berbagai tujuan seperti visualisasi, spesifikasi, konstruksi, dan dokumentasi sistem perangkat lunak. Sebagaimana seorang arsitek yang membuat blueprint untuk konstruksi bangunan, arsitek perangkat lunak menggunakan diagram UML untuk membantu programmer/developer dalam membangun perangkat lunak (Khairunnisa et al., 2024).

Dalam penlitian lain yang dilakukan oleh (Harlina et al., 2025) *Unified Modeling Language (UML)* merupakan pemodelan *Object Oriented* yang menganalogikan sebuah sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol – simbol yang cukup spesifik. Saat ini Sebagian besar para perancang sistem dalam menggambarkan informasi dengan memanfaatkan UML dengan tujuan utama untuk membantu tim proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat.

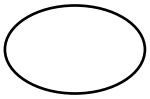
Lebih lanjut, pemodelan menggunakan UML juga sangat krusial untuk meminimalisir kesalahan logika pada saat proses pengembangan berlangsung. Menurut penelitian (Rizky Pangestu & Voutama, 2024), pemanfaatan UML berfungsi sebagai fondasi standar pemodelan yang memudahkan pengembang dalam mengidentifikasi kebutuhan

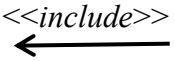
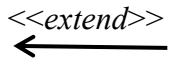
pengguna (*user requirement*), merancang relasi antardata, dan memetakan seluruh aktivitas operasional secara mendetail sebelum tahap pengkodean (*coding*) dilakukan. Rancangan visual yang komprehensif dari UML ini sangat dibutuhkan untuk memastikan agar sistem yang dibangun dapat terstruktur dengan baik, terintegrasi, dan sesuai dengan tujuan awal perancangan.

a. Use case diagram

Use case diagram adalah gambaran grafis dari beberapa atau semua actor, use case, dan interaksi diantaranya yang memperkenalkan suatu system, untuk di kembangkan dalam Use Case yang di sajikan terdapat empat proses utama yang di lakukan dalam sistem utama yang di lakukan dalam siste informasi monitoring yang akan di buat (Aldi Ramadani, 2025).

Tabel 2. 2 Simbol Use Case

No	Simbol	Nama	Keterangan
1		Aktor	Peran eksternal yang berinterkasi dengan sistem, biasanya di gambarkan sebagai aktor.
2		*Use Case*	Abstraksi antara sistem dan aktor
3		*Association*	Penghubung *use case* dengan aktor

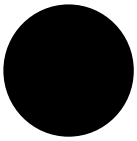
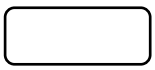
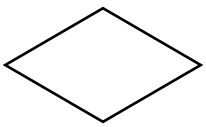
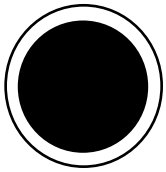
4		Generalisasi	Spesialisasi aktor untuk berhubungan dengan *use case*
5		*Include*	Simbol yang menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.
6		*Extend*	Simbol yang menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.

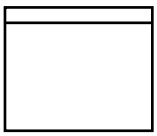
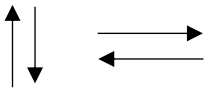
Sumber : (Permana & Utami, 2025)

b. Activity diagram

Activity diagram merupakan salah satu diagram UML yang digunakan untuk memodelkan alur aktivitas atau aliran kerja (workflow) dalam suatu sistem. Diagram ini memberikan gambaran mengenai tahapan – tahapan proses yang dilakukan pengguna maupun sistem, mulai dari awal hingga akhir (Ichsan Syah Putra & Danyl Mallisza, 2025).

Tabel 2. 3 Simbol Actvity Diagram

No	Simbol	Nama	Keterangan
1		Status Awal	Melambangkan titik permulaan dalam sebuah diagram aktivitas yang menunjukkan dimulainya proses.
2		Aktivitas	Menunjukkan suatu tindakan atau proses yang di lakukan oleh sistem, biasanya diawali dengan kata kerja.
3		*Decision*	Menyatakn titik percabangan dalam alur proses, dimana terdapat lebih dari satu opsi aktivitas yang dapat di pilih
4		*Join*	Mengilustrasikan proses penggabungan dua atau lebih aktivitas menjadi satu alur kegiatan.
5		Status Akhir	Merepresentasikan titik penyelesaian dalam diagram aktivitas yang menandakan

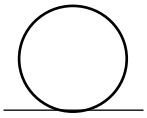
			berakhirnya proses sistem.
6		*Swimlane*	Menyediakan reprenstasi visual untuk membedakan peran atau tanggung jawab antar entitas dalam suatu organisasi berdasarkan aktivitas yang di lakukan
7		*Line connector*	Menunjukkan hubungan atau keterkaitan antara elemen – elemen dalam diagram alur aktivitas

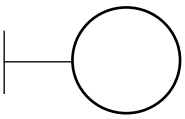
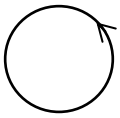
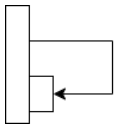
Sumber : (Fitriana et al., 2025)

c. Sequence diagram

Sequence diagram merupakan sebuah diagram yang mengambarkan kelakuan dari objek yang ada pada use case dengan cara mendeskripsikan waktu kejadian objek dan pesan yang akan dikirm dan diterima oleh antar objek (Elsa Rahmadani, 2025).

Tabel 2. 4 Sequence Diagram

No	Simbol	Nama	Keterangan
1		*Entity Class*	Simbol yang menggambarkan sistem sebagai landasan dalam

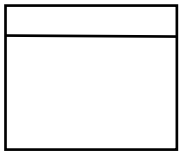
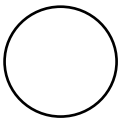
			Menyusun basis data, merepresentasikan proses utama dalam pemodelan data.
2		*Boundary Class*	Simbol yang menangani komunikasi antar lingkungan sistem, sebagai jembatan informasi antara sistem internal dan eksternal, sebagai jembatan informasi antara sistem internal dan eksternal
3		*Control Class*	Simbol yang bertanggung jawab terhadap class objek yang berisi logika, entitas pengelola operasi atau fungsi dalam sistem.
4		*Recursive*	Simbol yang melambangkan pesan untuk dirinya, menandakan pemanggilan metode internal dalam objek yang sama.

Sumber : (Syahputra, 2025)

d. Class diagram

Class Diagram menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi Keadaan (metode/fungsi) tersebut (Dhika Widiyanto Adhi & Anung Wahyu Adhi, 2025).

Tabel 2. 5 Simbol class diagram

No	Simbol	Nama	Keterangan
1		*Class*	Struktur utama untuk memodelkan objek dalam sistem.
2		*Interface*	Merupakan representasi antarmuka dalam pemrograman berorientasi objek, yang harus diimplementasikan oleh kelas.
3		*Association*	Hubungan umum antar kelas, sering disertai multiplicity untuk menunjukkan jumlah objek yang terlibat dalam relasi.
4		*Direct Association*	Menunjukkan hubungan antara satu kelas dengan kelas

			lainnya yang memanfaatkannya; hubungan ini sering disertai dengan multiplicity.
5		*Generalisasi*	Mengilustrasikan relasi pewarisan antar kelas, dari yang bersifat umum ke yang lebih spesifik (spesialisasi).
6		*Dependency*	Menandakan adanya ketergantungan satu kelas terhadap kelas lainnya dalam sistem.
7		*Aggregation*	Menjelaskan relasi bagian keseluruhan (whole-part) antara kelas-kelas dalam sistem.

Sumber : (Mendonca & Sulianta, 2024)

B. Kajian Empiris

Dalam Menyusun penilitian ini, penulis merujuk pada beberapa jurnal yang memiliki keterkaitan dengan topik yang di bahas. Oleh karena itu, penulis mengakat tiga hasil penelitian terdahulu yang dianggap relevan sebagai landasan referensi:

1. Pada penilitian yang di lakukan oleh (Susanti et al., 2023) dengan judul "*Absensi Mahasiswa Berbasis Pengenalan Wajah Menggunakan*

Algortima Yolov5". Penelitian ini mengembangkan sistem pengenalan wajah secara otomatis guna meningkatkan keamanan dan meminimalisir kecurangan. Hasil pengujian menunjukkan kecepatan tinggi dan akurasi tinggi yang baik, sehingga sistem ini sangat relevan untuk diadaptasikan pada sistem kontrol akses otomatis.

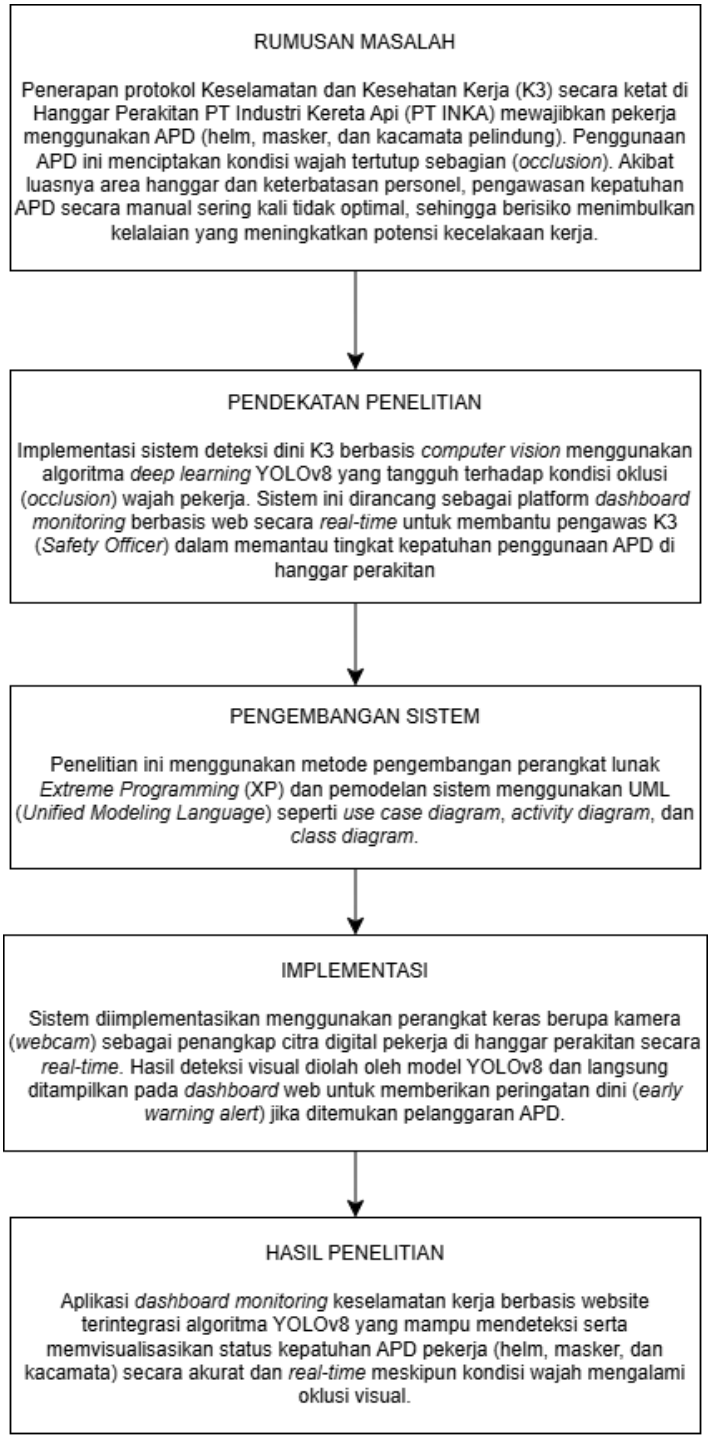
2. Penelitian serupa yang mengeksplorasi kondisi hambatan pada wajah di lakukan dalam penelitian "*Sistem Pengenalan Wajah Real – Time Menggunakan YOLOv7 untuk Akses Gedung TVRI Palembang Berbasis Web*" (Kyara et al., 2025). Penelitian ini berfokus pada implementasi algoritma *Deep Learning* dari keluarga YOLO untuk memberikan izin kontrol akses masuk ke dalam gedung berkeamanan khusus. Berdasarkan hasil pelatihan (*training*) sistem pada dataset wajah, model tersebut menunjukkan performa deteksi yang sangat optimal dengan nilai *Precision* mencapai 0,941 dan *Recall* sebesar 1. Hal ini membuktikan bahwa algoritma YOLO menawarkan keseimbangan terbaik antara kecepatan komputasi *real – time* dan tingkat keamanan fisik pada sebuah bangunan.
3. Penelitian terkait penanganan variasi wajah (*occlusion*) diuji dalam "*Rancang Bangun Sistem Kehadiran Secara Real Time Menggunakan Face Recognition Dengan Metode SSD di SMK"* yang di publikasikan oleh (Azzahra & Ananda, 2024). Penelitian ini mengevaluasi ketangguhan sistem *biometric* wajah terhadap kondisi *face occlusion,* di mana wajah pengguna sebagian terhalang oleh penggunaan masker

atau aksesoris lain di area kepala. Hasil pengujian menunjukkan sistem mampu menagani kondisi penghalang tersebut dengan tingkat akurasi pengenalan wajah mencapai 90 % dan tingkat efisiensi waktu deteksi sebesar 95,8%. Hal ini membuktikkan bahwa pendekatan *deep learning* sangat andal dalam mengenali fitur wajah pada kondisi lingkungan kerja yang dinamis.

Dari beberapa sumber literatur di atas, maka penggunaan metode yang tepat dalam proses Pembangunan sistem keamanan sangat diperlukan. Dalam proses pengembangan sistem ini, penulisan menerapkan pemodelan arsitektur *Deep learning* (YOLO) untuk membangun sistem kontrol akses otomatis pada pintu hanggar perakitan di PT. Industri Kereta Api (INKA). Perbedaan utama penelitian ini di bandingkan penelitian sebelumnya terletak pada integrasi sistem *face recognition* yang di khususkan pada penanganan kondisi *occlusion* secara ekstrem, dimana para pekerja di wajibkan menggunakan Alat pelindung diri (APD) seperti helm *safety*, kacamata, atau masker industry pelindung debu. Walaupun wajah tertutup oleh APD. sistem di harapkan tetap mampu medeteksi fitur mata dan struktur tulang wajah secara presisi untuk menggerakkan perangkat keras kunci *solenoid* pada pintu hanggar PT. INKA.

C. Kerangka Berfikir

Penjelasanan berikut menyajikan kerangka berpikir yang di gunakan dalam pelaksanaan penelitian ini.



Gambar 2. 1 Kerangka Berfikir