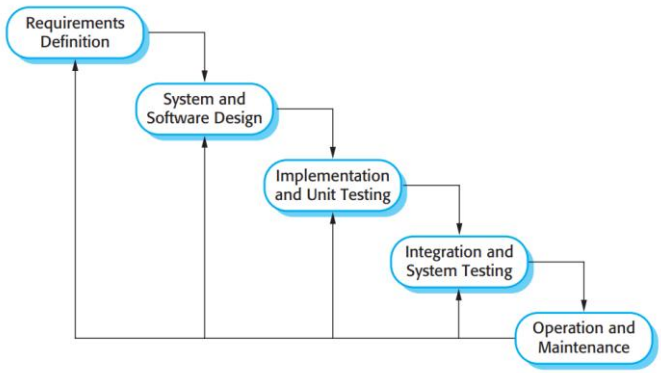
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Berlandaskan pada fokus dan tujuan penelitian yang telah dirumuskan sebelumnya, penelitian ini dimaksudkan untuk mengembangkan sistem deteksi *road accident* yaitu kecelakaan lalu lintas dan tindakan kriminal bersenjata berbasis *web*. Sistem ini memanfaatkan teknologi *deep learning* dengan model YOLOv11 untuk mendeteksi kejadian secara *real-time* dan terintegrasi *web monitoring* dengan sistem *alert* yang bertujuan meningkatkan respons unit tanggap darurat khususnya kepolisian.

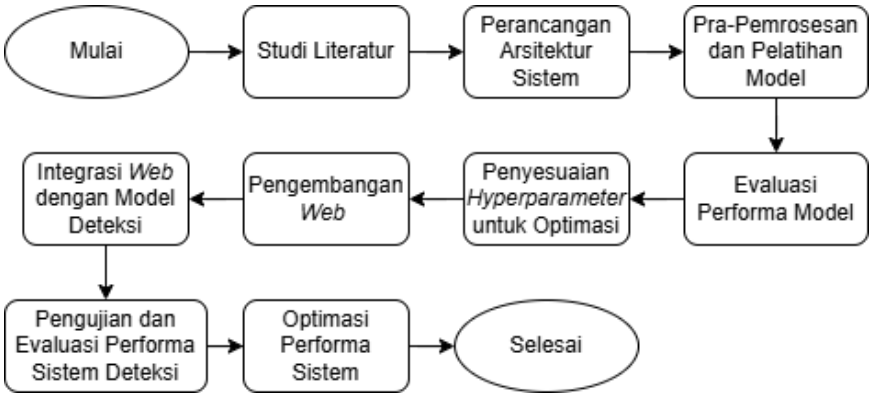
Penelitian ini menggunakan metode *Research and Development* (R&D) dengan model pengembangan yang digunakan adalah pendekatan *Waterfall*. Pendekatan *Waterfall* merupakan model pengembangan perangkat lunak klasik yang menerapkan alur kerja berurutan yang dimulai dari analisis kebutuhan, perancangan, implementasi dan tes unit, integrasi dan tes sistem, serta pemeliharaan. Model ini dipilih karena memberikan kerangka kerja yang terstruktur dan sistematik sehingga meminimalkan risiko kesalahan pada setiap tahap proses pengembangan (Sommerville, 2011). Gambar 3.1 berikut adalah tahapan penelitian R&D dengan model pendekatan *Waterfall*.



Gambar 3.1 Model Pengembangan *Waterfall*

Sumber: Sommerville (2011)

Adapun Gambar 3.2 berikut adalah langkah-langkah penelitian model *waterfall* yang merujuk pada tahapan *Research and Development* (R&D) untuk pengembangan sistem deteksi kecelakaan lalu lintas dan tindakan kriminal bersenjata berbasis *web*.



Gambar 3.2 Prosedur Penelitian Berdasar Model Pengembangan *Waterfall*

1. Analisis Kebutuhan

Pada tahap analisis kebutuhan, dilakukan identifikasi mendalam terhadap permasalahan kecelakaan lalu lintas dan tindakan kriminal bersenjata yang terjadi di lapangan serta keterbatasan sistem deteksi yang ada saat ini. Penelitian ini mengembangkan sistem deteksi yang mampu menghadirkan solusi terhadap tantangan keamanan di wilayah publik untuk menjawab permasalahan tersebut. Salah satu kebutuhan mendasar dalam membuat sistem ini adalah membangun basis data yang mampu merepresentasikan variasi kompleks kejadian *road accident*. Maka dari itu, dilakukan strategi pengumpulan *dataset* dari sumber publik dan pengambilan gambar secara langsung yang menggambarkan kecelakaan, penggunaan senjata, dan kondisi lalu lintas normal.

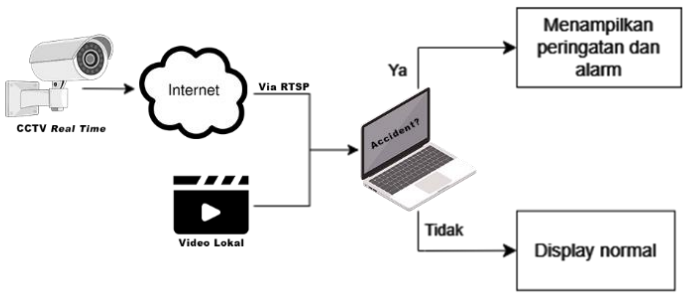
Proses analisis dilanjutkan dengan penentuan spesifikasi teknis untuk model deteksi berbasis YOLOv11 seperti resolusi *input* gambar, konfigurasi *hyperparameter* pelatihan, serta pemilihan metrik evaluasi seperti *precision*, *recall*, dan *mean average precision* (mAP). Analisis juga mempertimbangkan tantangan teknis yang mungkin muncul, seperti variasi kondisi pencahayaan,

sudut pengambilan gambar, dan gangguan visual lainnya yang dapat berdampak pada akurasi deteksi objek.

Pada aspek kebutuhan fungsional, sistem yang dikembangkan harus mampu mendeteksi secara *real-time* kejadian kecelakaan lalu lintas dan tindakan kriminal bersenjata dengan tingkat akurasi yang tinggi. Sistem dirancang untuk memiliki modul pengolahan citra yang terintegrasi dengan model YOLOv11 sehingga setiap deteksi akan langsung diteruskan ke *platform web* yang dikembangkan dengan *framework* Flask. Selain itu, antarmuka pengguna (UI) harus intuitif dan responsif yang memungkinkan petugas tanggap darurat untuk memantau secara *live*, menerima *alert*, dan mengakses *log* data untuk analisis lebih lanjut. Dengan demikian, analisis kebutuhan ini menjadi dasar yang kuat untuk merancang sistem yang efisien dan efektif dalam meningkatkan respon terhadap kecelakaan dan tindakan kriminal di jalan raya.

2. Perancangan Sistem

Pada tahap perancangan sistem dilakukan perumusan arsitektur keseluruhan yang mencakup identifikasi komponen-komponen utama serta hubungan interaksi antar modul. Gambar 3.3 menunjukkan arsitektur sistem deteksi, di mana terdapat dua sumber video utama yang diintegrasikan, yaitu CCTV *real-time* melalui RTSP dan video lokal. Setelah melalui proses analisis oleh model deteksi, sistem akan memutuskan apakah perlu menampilkan peringatan dan alarm (jika terdeteksi kecelakaan) atau menampilkan tampilan normal (jika tidak ada kecelakaan).



Gambar 3.3 Arsitektur Sistem Deteksi

Selanjutnya, perancangan mendetail difokuskan pada spesifikasi masing-masing modul. Pada sisi *backend*, sistem mengintegrasikan model *deep learning* YOLOv11 dengan *framework* Flask untuk menangani pengolahan dan penyimpanan data deteksi. Modul ini dirancang untuk memastikan *latency* yang rendah dan keandalan tinggi dalam mengirimkan data ke *frontend*. Sementara itu, perancangan antarmuka pengguna (UI) dilakukan untuk menciptakan tampilan *monitoring* yang intuitif, di mana petugas dapat dengan mudah mengakses *live streaming*, notifikasi, dan *log* data kejadian. Dengan demikian, seluruh komponen sistem dapat saling terhubung untuk mendukung proses deteksi dan respons cepat terhadap kecelakaan atau tindakan kriminal di jalan.

3. Implementasi

Tahap implementasi dimulai dengan pengembangan modul pengolahan citra dan pelatihan model YOLOv11. Pada tahap ini, dilakukan pra-pemrosesan *dataset* yang mencakup pengumpulan *dataset*, penyesuaian *dataset*, dan anotasi sehingga data sesuai dengan format *input* model. Selanjutnya, *dataset* yang telah diproses dibagi menjadi data pelatihan, validasi, dan pengujian. Proses pelatihan dijalankan menggunakan GPU T4 Tesla pada Google Colaboratory guna mengoptimalkan waktu komputasi dan mempercepat konvergensi model.

Setelah model dilatih, dilakukan evaluasi performa menggunakan metrik evaluasi standar seperti *precision*, *recall*, dan *mean average precision* (mAP). Metrik *precision* mengukur seberapa besar proporsi deteksi yang benar dari keseluruhan deteksi, sedangkan *recall* mengukur seberapa banyak kejadian yang berhasil terdeteksi dari seluruh kejadian yang sebenarnya ada. Adapun nilai mAP mencerminkan performa deteksi keseluruhan pada berbagai *threshold Intersection over Union* (IoU). Hasil evaluasi ini menjadi dasar untuk menilai tingkat keandalan model YOLOv11 dalam mendeteksi kejadian kecelakaan atau tindakan kriminal serta memberikan umpan balik untuk melakukan penyesuaian *hyperparameter* apabila diperlukan.

4. Integrasi dan Uji Sistem

Setelah model YOLOv11 mencapai performa yang diharapkan, dilanjutkan dengan integrasi model ke dalam sistem berbasis *web* menggunakan *framework* Flask. Pada tahap ini, dibuat API yang menghubungkan proses inferensi model dengan modul *backend* sehingga hasil deteksi dapat dikirim secara *real-time* ke antarmuka pengguna. Pengembangan *frontend* dilakukan dengan memperhatikan desain yang intuitif dan responsif sehingga petugas dapat dengan mudah memantau *live streaming*, menerima *alert*, serta mengakses data log untuk analisis lebih lanjut.

Pengujian sistem dilakukan dengan dua skenario utama, yaitu video lokal dan CCTV *real-time*. Hasil pengujian ini menjadi dasar untuk melakukan perbaikan dan optimasi sehingga sistem yang dikembangkan dapat beroperasi secara efektif dan andal dalam mendeteksi kecelakaan lalu lintas serta tindakan kriminal bersenjata.

5. Pemeliharaan

Pada tahap pemeliharaan, dilakukan analisis menyeluruh terhadap kinerja sistem setelah implementasi untuk memastikan bahwa sistem berjalan dengan optimal. Evaluasi ini mencakup peninjauan metrik evaluasi yaitu akurasi, *frame rate*, kecepatan deteksi, latensi, dan respons sistem *alert* berdasarkan hasil uji sistem yang telah dilakukan untuk mengidentifikasi potensi kelemahan sistem yang mungkin muncul saat beroperasi dalam kondisi nyata di lapangan.

Hasil evaluasi tersebut digunakan sebagai dasar untuk menentukan area yang memerlukan perbaikan dan penyempurnaan lebih lanjut. Optimalisasi performa sistem melalui pembaruan dan penyesuaian yang tepat diharapkan dapat membuat sistem menjadi lebih responsif dan efektif dalam menghadapi situasi darurat.

3.2 Proses Perancangan dan Pengembangan

3.2.1 Pra-Pemrosesan *Dataset*

1. Pengumpulan *Dataset*

Pada tahap pengumpulan *dataset*, data dikumpulkan dari dua sumber utama, yaitu *dataset* publik dan data yang diambil langsung dari lapangan. Dari *dataset* publik, kaggle digunakan untuk mendapatkan sampel gambar kecelakaan kendaraan dan penggunaan senjata tajam, sedangkan youtube menyediakan kumpulan video kecelakaan yang kemudian diekstrak menjadi gambar. Dengan menggunakan *dataset* publik ini, variasi kejadian dalam berbagai kondisi dapat dimanfaatkan untuk meningkatkan kemampuan model dalam mengenali pola kecelakaan dan tindakan kriminal dengan lebih akurat.

Selain *dataset* publik, data lapangan juga dikumpulkan untuk meningkatkan realisme dan representativitas sistem deteksi. Dalam hal ini, dilakukan pengambilan video secara langsung di tiga jembatan penyeberangan orang (JPO) yang tersebar di beberapa ruas jalan Kota Bandung untuk mengumpulkan data lalu lintas yaitu JPO Jalan Merdeka, Jalan Ir. Juanda, dan Jalan Pasir Kaliki. Pengambilan data dilakukan dengan memanfaatkan sudut pandang atas untuk memperoleh visibilitas optimal terhadap pergerakan kendaraan. Ketiga lokasi dipilih untuk memberikan variasi spasial dalam *dataset* yang mencakup perbedaan konfigurasi jalan seperti variasi jumlah lajur, keberadaan median jalan, dan kedekatan dengan fasilitas publik. Selain itu, untuk melengkapi *dataset* deteksi senjata dalam konteks kriminalitas di jalan, dilakukan juga pengambilan data simulasi yang menampilkan subjek memegang pisau dalam berbagai posisi dan sudut. *Dataset* simulasi senjata tajam ini diambil dalam kondisi terkontrol dengan mempertimbangkan variasi pencahayaan, jarak, dan posisi pemegang senjata untuk memaksimalkan kemampuan sistem dalam mendeteksi potensi tindak kriminal. Penggabungan *dataset* publik dan data lapangan dalam sistem yang dikembangkan ini diharapkan mampu mendeteksi kecelakaan dan tindakan kriminal bersenjata dengan lebih presisi serta beradaptasi terhadap berbagai kondisi lingkungan.

2. Penyesuaian *Dataset*

Pada tahap penyesuaian *dataset*, dilakukan serangkaian proses untuk memastikan bahwa data memiliki format yang seragam dan sesuai dengan kebutuhan model deteksi YOLOv11. Langkah pertama adalah *auto orient*, yaitu penyesuaian orientasi gambar secara otomatis agar seluruh *dataset* memiliki keselarasan dalam sudut pandang. Dengan demikian, model dapat mengenali objek secara konsisten tanpa terganggu oleh perbedaan orientasi yang mungkin muncul akibat variasi sumber data. Proses ini sangat penting untuk meningkatkan akurasi deteksi, terutama ketika *dataset* dikumpulkan dari berbagai sumber dengan kondisi pengambilan gambar yang berbeda.

Selanjutnya, dilakukan proses *resize* gambar untuk menyesuaikan seluruh *dataset* ke resolusi 640x640 piksel. Penyesuaian ini dilakukan agar ukuran input sesuai dengan arsitektur model YOLOv11 sehingga dapat meningkatkan efisiensi komputasi serta keakuratan deteksi. Proses *auto orient* dan *resize* ini dilakukan secara otomatis menggunakan *platform* Roboflow yang menyediakan alat untuk mengoptimalkan *dataset* sebelum digunakan dalam proses pelatihan model.

Untuk menambah keanekaragaman data dan meningkatkan kemampuan model dalam menghadapi variasi kondisi visual, dilakukan juga augmentasi *dataset*. Tiga teknik augmentasi yang diterapkan adalah horizontal *flipping*, *brightness,* dan *grayscale*. Horizontal *flipping* memberikan variasi orientasi gambar sehingga model dapat mengenali objek meskipun tampilannya terbalik secara horizontal. Penyesuaian *brightness* diaplikasikan untuk mensimulasikan perubahan kondisi pencahayaan, baik dalam situasi terang maupun redup yang sering terjadi pada rekaman CCTV dan video lapangan. Sementara itu, *grayscale* yaitu mengubah gambar berwarna menjadi hitam-putih sehingga model dapat mengenali fitur tanpa bergantung pada informasi warna. Proses penyesuaian ini dapat menjadikan *dataset* yang dihasilkan lebih seragam dan kaya variasi sehingga mendukung performa model dalam mendeteksi kejadian kecelakaan dan tindakan kriminal bersenjata secara *real-time*.

3. Anotasi *Dataset*

Pada tahap Anotasi *dataset*, dilakukan proses pelabelan menggunakan *platform* Roboflow dengan memberikan label pada setiap gambar sesuai dengan kelas yang telah ditentukan, yaitu bukan kecelakaan, kecelakaan, dan senjata. Gambar-gambar yang telah dikumpulkan diberikan anotasi berupa *bounding box* untuk masing-masing kelas sehingga setiap objek yang muncul dapat diidentifikasi dengan jelas dan akurat. Proses ini sangat penting untuk memastikan bahwa data yang digunakan dalam pelatihan model YOLOv11 memiliki label yang konsisten dan tepat sehingga akan mendukung peningkatan akurasi deteksi.

Setelah proses anotasi selesai, *dataset* yang telah diberi label diverifikasi untuk memastikan kesesuaian dan keakuratan setiap anotasi. Verifikasi dilakukan dengan memeriksa kembali penempatan *bounding box* dan kebenaran label pada masing-masing gambar serta melakukan penyesuaian bila ditemukan inkonsistensi.

3.2.2 Pengembangan dan *Training* Model

Setelah tahapan pra-pemrosesan selesai, selanjutnya dilakukan *split dataset* sebagai bagian dari tahap pengembangan dan *training* model. Data yang telah dianotasi dibagi secara acak ke dalam tiga subset untuk mendukung proses pelatihan dan evaluasi model secara optimal. Pembagian *dataset* ini sangat penting karena membantu memastikan bahwa model dapat belajar dari berbagai variasi data secara menyeluruh dan menghindari *overfitting*. Dengan memisahkan data untuk pelatihan, validasi, dan pengujian, model diuji pada data yang tidak terlihat selama proses pelatihan sehingga memberikan gambaran yang lebih akurat mengenai kemampuan generalisasinya. Hal ini krusial untuk mengidentifikasi kelemahan model dan melakukan penyesuaian *hyperparameter* yang tepat guna meningkatkan performa deteksi secara keseluruhan. Pembagian *dataset* ini dilakukan dengan proporsi seperti ditunjukkan pada Tabel 3.1.

Tabel 3.1 Proporsi Pembagian *Dataset*

Data Pelatihan (*Train*)	Data Validasi (Val)	Data Pengujian (*Test*)
85%	10%	5%

Pada Tabel 3.1, Proporsi 85% untuk data pelatihan memungkinkan model untuk belajar dari variasi data yang luas, sedangkan 10% data validasi digunakan untuk mengoptimalkan *hyperparameter* dan mencegah *overfitting* selama proses pelatihan. Sementara itu, data pengujian 5% berfungsi untuk mengukur kemampuan generalisasi model pada data yang belum pernah dilihat sebelumnya. Pemilihan proporsi 85:10:5 ini bertujuan untuk mengefektifkan proses *training* dan evaluasi seperti yang dilakukan pada penelitian A. Sharma dkk. (2024) yang memiliki karakteristik jumlah *dataset* yang tidak terlalu besar yaitu 2348 *dataset*.

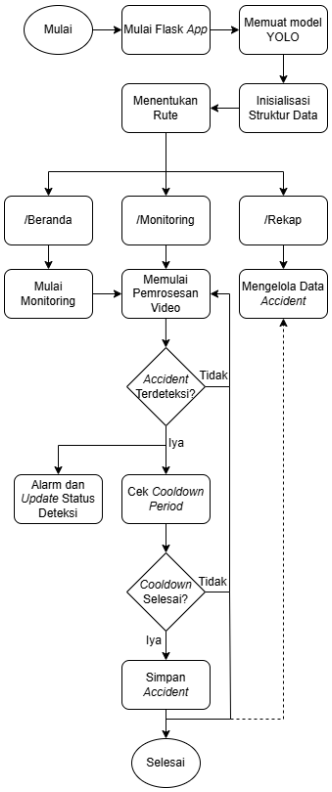
Pada tahap pengembangan dan *training* model ini, YOLOv11 dikembangkan untuk mendeteksi objek berdasarkan *dataset* yang telah dipersiapkan. Model ini dipilih karena kemampuannya dalam mendeteksi objek secara *real-time* dengan akurasi tinggi yang sangat relevan untuk mendukung sistem deteksi kecelakaan dan tindakan kriminal bersenjata. Proses pengembangan mencakup konfigurasi arsitektur model dan penyusunan *pipeline training* menggunakan *framework deep learning* dengan *dataset* yang telah dianotasi dalam kategori bukan kecelakaan, kecelakaan, dan senjata.

3.2.3 Evaluasi Performa Model Deteksi

Pada tahap evaluasi performa model, pengukuran efektivitas deteksi dilakukan dengan menggunakan metrik-metrik standar seperti *precision*, *recall*, dan *mean average precision* (mAP). Data validasi dan pengujian yang sebelumnya telah dipisahkan digunakan untuk mengevaluasi kemampuan model dalam mengenali objek secara akurat pada gambar yang belum pernah dilihat selama proses pelatihan. Proses evaluasi ini melibatkan analisis terhadap jumlah *true positives*, *false positives*, dan *false negatives* untuk masing-masing kelas sehingga dapat diperoleh gambaran jelas mengenai kelebihan dan kelemahan model dalam mendeteksi kategori bukan kecelakaan, kecelakaan, dan senjata. Hasil evaluasi digunakan sebagai dasar untuk menyesuaikan *hyperparameter* dan melakukan iterasi perbaikan model jika diperlukan.

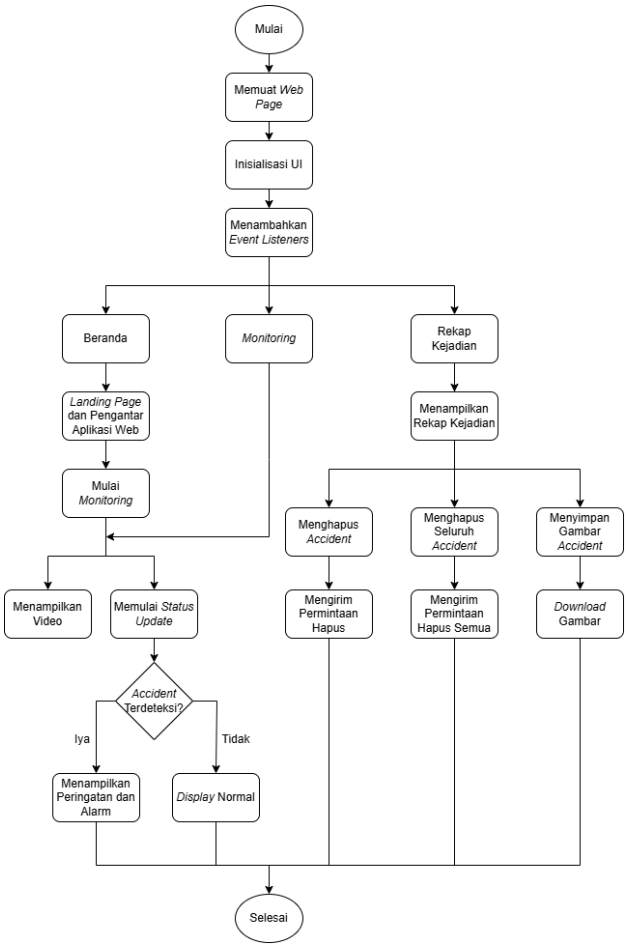
3.2.4 Pengembangan *Web* dan Integrasi dengan Model Deteksi

Sistem ini dirancang dengan arsitektur yang terbagi menjadi dua komponen utama, yaitu *backend* dan *frontend* untuk memastikan pemisahan tugas yang efisien antara pemrosesan data di sisi server dan interaksi pengguna di sisi klien. *Backend* berperan sebagai inti sistem yang berjalan di *server* di mana mencakup aplikasi Flask, model YOLOv11, dan penyimpanan data *accident*. Dalam pengembangan *web* dan integrasi ini, *backend* mengintegrasikan model YOLOv11 yang telah dilatih ke dalam aplikasi Flask, dimulai dengan inisialisasi struktur data dan pemuatan model untuk mendukung inferensi secara *real-time*. Flask berfungsi sebagai *framework* yang mengelola *routes* dan *requests* dari pengguna. Pada Gambar 3.4, alur *backend* sistem dimulai dari memulai aplikasi Flask, memuat model YOLO, kemudian memproses aliran video yang diterima. Jika terdeteksi accident, sistem akan secara otomatis menyimpan data tersebut untuk keperluan analisis dan pelaporan lebih lanjut.



Gambar 3.4 Diagram Alir *Backend*

Pada sisi lain, *frontend* seperti digambarkan pada Gambar 3.5, dirancang untuk berinteraksi langsung dengan pengguna melalui *browser*. Proses dimulai dengan memuat halaman *web* yang berisi antarmuka pengguna (UI) dan *event listeners* untuk mengelola interaksi. Pengguna dapat memulai *monitoring* dengan menekan tombol yang tersedia yang selanjutnya memicu alur *backend* untuk memproses video. Halaman *web* akan menampilkan status deteksi secara *real-time*, termasuk menandai jika terdeteksi *accident*. Ketika model mendeteksi *accident*, sistem akan menampilkan peringatan disertai alarm dan menyimpan gambar tersebut di rekap kejadian. Pengguna dapat mengakses halaman rekap kejadian untuk meninjau daftar *accident*, menghapus data, dan atau mengunduh gambar untuk keperluan pelaporan.



Gambar 3.5 Diagram Alir *Frontend*

3.2.5 Uji Integrasi *Backend* dan *Frontend*

Uji integrasi *backend* dan *frontend* dilakukan untuk memastikan bahwa kedua komponen sistem, yaitu *backend* dan *frontend* dapat bekerja sama dengan baik. Pengujian ini menggunakan metode *black box* yang berfokus pada *input* dan *output* sistem tanpa memperhatikan detail internal kode.

Setiap fitur diuji dengan skenario yang mencakup input pengguna, aksi komunikasi antara *frontend* dan *backend*, serta *output* yang diharapkan. Pengujian dilakukan dengan menjalankan sistem secara keseluruhan untuk memverifikasi apakah semua fungsi berjalan sesuai spesifikasi. Tabel 3.2 berikut menunjukkan skenario pengujian untuk setiap fitur yang melibatkan integrasi *backend* dan *frontend*.

Tabel 3.2 Skenario Uji Integrasi *Backend* dan *Frontend*

No	Fitur	Skenario	*Input*	Aksi	*Output* yang diharapkan
1	Halaman Beranda	Pengguna navigasi ke halaman Beranda	Klik “Beranda”	*Frontend* meminta halaman dari *backend*	Halaman Beranda ditampilkan
2	Halaman *Monitoring*	Pengguna navigasi ke halaman *Monitoring*	Klik “Monitoring”	*Frontend* meminta halaman dari *backend* dan meminta *stream* dari *backend*	Halaman *monitoring* ditampilkan dan video ditampilkan dengan lancar
3	Halaman Rekap kejadian	Pengguna navigasi ke halaman	Klik “Rekap Kejadian”	*Frontend* meminta halaman	Halaman Rekap kejadian

No	Fitur	Skenario	*Input*	Aksi	*Output* yang diharapkan
		Rekap Kejadian		dari *backend* dan meminta data historis	ditampilkan dan data kejadian ditampilkan di tabel
4	Mulai *Monitoring*	Pengguna mengakses halaman Beranda dan *button* Mulai Monitorig	Klik “Mulai Monitoring”	*Frontend* meminta halaman dari *backend* dan meminta *stream* dari *backend*	Halaman *monitoring* ditampilkan dan video ditampilkan dengan lancar
5	Deteksi Objek *Real-Time*	Video berjalan dengan deteksi	Video *streaming* aktif	*Backend* mengirim hasil deteksi ke *frontend*	*Bounding box* dan label muncul di video
6	Notifikasi *Alert*	Kejadian terdeteksi	Kecelakaan/ senjata terdeteksi	*Backend* mengirim sinyal *alert*	*Alert* suara diputar, panel status merah
7	Status Deteksi	Status diperbarui *real-time*	Video streaming aktif	*Backend* mengirim status via API	Panel status berubah (hijau/merah) sesuai deteksi

No	Fitur	Skenario	*Input*	Aksi	*Output* yang diharapkan
8	Hapus Gambar *Accident*	Pengguna menghapus satu gambar	Klik *button* "Hapus" pada gambar	*Frontend* mengirim permintaan hapus ke *backend*	Gambar dihapus dari halaman dan sistem
9	Hapus Seluruh Gambar *Accident*	Pengguna menghapus semua gambar	Klik "Hapus Semua Kejadian"	*Frontend* mengirim permintaan hapus semua gambar ke *backend*	Semua gambar dihapus dari halaman dan sistem
10	*Download* Gambar *Accident*	Pengguna mengunduh gambar	Klik ikon "*Download*"	*Frontend* meminta file dari *backend*	Gambar diunduh ke perangkat pengguna

3.2.6 Pengujian Sistem

Pada tahap pengujian sistem, dilakukan evaluasi komprehensif terhadap integrasi model YOLOv11 ke dalam platform web dengan fokus pada lima aspek utama sesuai rumusan masalah, yaitu akurasi deteksi, *frame rate* (FPS), kecepatan deteksi, latensi sistem, dan respon sistem *alert*. Pengujian ini dirancang untuk memberikan gambaran menyeluruh tentang kemampuan sistem dalam mendeteksi kejadian kecelakaan lalu lintas dan keberadaan senjata secara *real-time* pada implementasi praktis.

1. Desain Pengujian

Sistem diuji menggunakan dua metode pengujian yaitu menggunakan video lokal dan CCTV *real-time*. Pada metode Video Lokal, sistem secara langsung

memproses file video yang disimpan dalam penyimpanan lokal, memberikan dasar evaluasi dalam kondisi optimal tanpa pengaruh latensi jaringan atau degradasi kualitas video yang biasa terjadi pada *streaming*. Sementara itu, metode CCTV *real-time* mensimulasikan kondisi penggunaan realistis di mana video diputar pada perangkat kedua (laptop) dan direkam secara *real-time* oleh kamera CCTV yang terhubung ke sistem deteksi. Hal ini memungkinkan evaluasi sistem dalam kondisi yang lebih menantang dengan adanya faktor tambahan seperti latensi jaringan, potensi degradasi kualitas video, dan variabilitas kondisi pencahayaan.

Untuk setiap metode pengujian, digunakan *dataset* video yang terdiri dari 15 video kecelakaan lalu lintas, 15 video yang menampilkan senjata, dan 15 video non-accident (kondisi normal tanpa kejadian penting). Total *dataset* pengujian mencakup 45 video yang bervariasi dalam kondisi pencahayaan, sudut kamera, dan kompleksitas *scene* untuk memastikan evaluasi yang komprehensif.

2. Metrik Pengukuran

a. Akurasi Deteksi

Akurasi deteksi diukur dengan membandingkan hasil deteksi sistem dengan *ground truth* dari *dataset*. Akurasi dihitung menggunakan rumus:

$$\text{Akurasi} = \frac{\text{Jumlah deteksi benar}}{\text{Total Video}} x100\% \quad \text{..... (2.3)}$$

Pengukuran akurasi dilakukan untuk setiap kelas (Bukan Kecelakaan, kecelakaan, dan Senjata) secara terpisah dan juga untuk keseluruhan *dataset*. Pendekatan ini memungkinkan evaluasi yang lebih terperinci tentang kinerja model pada berbagai jenis kejadian.

b. *Frame Rate* (FPS)

Frame rate diukur untuk mengevaluasi kemampuan sistem dalam memproses video secara *real-time*. Metrik ini dikalkulasi dengan menghitung jumlah *frame* yang diproses per detik menggunakan formula berikut:

$$FPS = \frac{1}{current\_time - prev\_frame\_time} \quad \text{..... (2.4)}$$

Di mana *current_time* adalah waktu selesainya pemrosesan *frame* saat ini dan *prev_frame_time* adalah waktu selesainya pemrosesan *frame* sebelumnya. Nilai FPS yang lebih tinggi menunjukkan pemrosesan yang lebih cepat dan responsif.

c. Kecepatan Deteksi

Kecepatan deteksi mengukur waktu yang dibutuhkan model untuk melakukan inferensi pada satu *frame*. Metrik ini dihitung dengan mencatat waktu mulai dan selesai inferensi model yang dirumuskan sebagai berikut:

$$Detection\ Time = inference\_end\_time - inference\_start\_time \quad \ldots. \quad (2.5)$$

d. Latensi Sistem

Pengukuran latensi sistem dilakukan untuk mengevaluasi responsivitas keseluruhan sistem deteksi. Sistem mencatat beberapa jenis latensi, diantaranya yaitu:

Network Latency: Waktu yang dibutuhkan untuk mendapatkan *frame* dari sumber video RTSP, dihitung dengan menggunakan rumus:

$$Network\ Latency = after\_capture - before\_capture \quad \ldots\ldots\ldots \quad (2.6)$$

Processing Latency: Waktu yang dibutuhkan untuk memproses *frame* dari saat perekaman hingga selesai diproses, dihitung dengan menggunakan rumus:

$$Processing\ Latency = processing\_end\_time - capture\_time \quad \ldots \quad (2.7)$$

Total Latency: Gabungan dari latensi jaringan dan latensi pemrosesan, dihitung dengan menggunakan rumus:

$$Total\ Latency = Processing\ Latency + Network\ Latency \quad \ldots\ldots. \quad (2.8)$$

e. Sistem Alert

Pengujian respons sistem *alert* dilakukan untuk melihat apakah sistem memberikan notifikasi suara dan menyimpan bukti ketika kejadian terdeteksi. Sistem *alert* suara diimplementasikan menggunakan *file audio* yang diputar ketika kejadian terdeteksi. *Alert* suara diaktifkan setiap kali

sistem mendeteksi kejadian sehingga dapat memberikan peringatan untuk setiap *frame* yang terdeteksi mengandung kecelakaan atau senjata.

3.3 Lingkungan Komputasi

Lingkungan komputasi yang digunakan dalam penelitian ini mencakup kombinasi perangkat keras dan perangkat lunak untuk mendukung seluruh proses mulai dari pengembangan, pelatihan, hingga *deployment* sistem. Pengembangan, *deployment*, dan pengujian sistem dilakukan menggunakan laptop dengan spesifikasi CPU AMD Ryzen 5 5500U dengan Radeon Graphics, RAM 16 GB, dan SSD 512 GB, serta IP CCTV V380 Pro model Y33. Sementara itu, untuk pelatihan model YOLOv11 yang memerlukan perangkat komputasi tinggi, digunakan layanan Google Colaboratory dengan dukungan GPU NVIDIA T4 Tesla, RAM 12,7 GB, dan total memori 107,7 GB yang memungkinkan proses *training* hingga 100 *epoch* dilakukan dengan lebih cepat dan efisien.

Selain perangkat keras, lingkungan perangkat lunak juga sangat mendukung kelancaran penelitian. Sistem dikembangkan menggunakan bahasa pemrograman Python dan memanfaatkan pustaka-pustaka penting seperti OpenCV untuk pemrosesan citra, Torch dan Ultralytics YOLO untuk inferensi model, serta Flask untuk pembangunan aplikasi *web*. Versi pustaka dan dependensi lainnya disesuaikan guna menjamin kompatibilitas dan kestabilan selama proses *training* dan *deployment*.