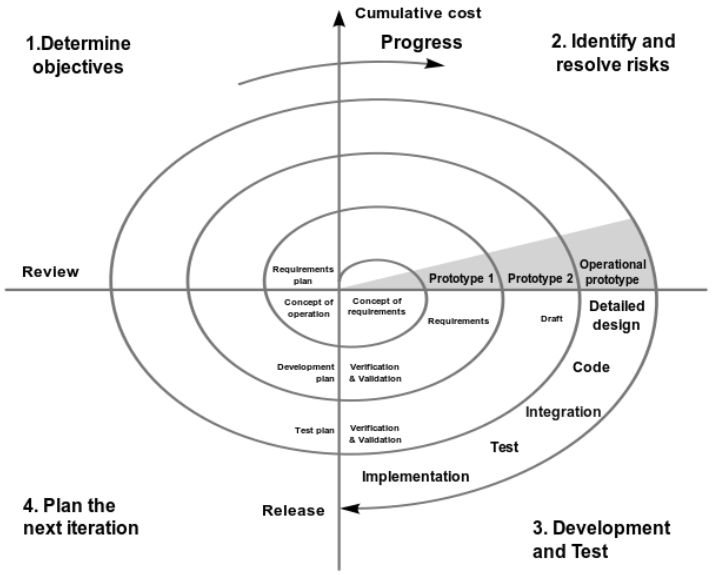
BAB 3
METODOLOGI PENELITIAN

3.1 Desain Penelitian

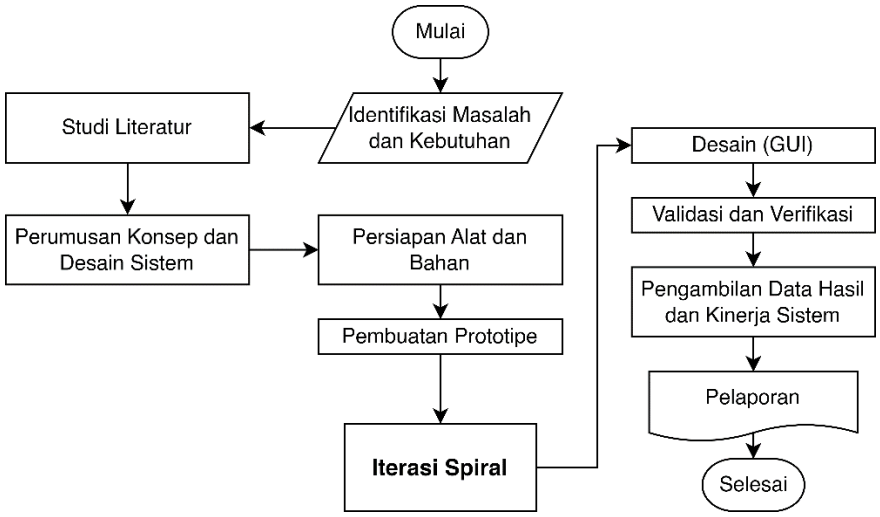
Berdasarkan tujuan penelitian yang telah dirumuskan, penelitian ini difokuskan untuk mengembangkan dan membuka potensi penerapan *computer vision* pada pembelajaran. *Computer vision* yang diterapkan berfungsi untuk melakukan rekognisi gestur tangan. Penelitian ini melibatkan penyelesaian pembuatan dan pengembangan sistem, serta antarmuka dari aplikasi, sesuai dengan rumusan konsep dan desain yang telah ditetapkan. Hasil akhir yang dicapai menghasilkan sebuah aplikasi yang dapat diimplementasikan dalam pembelajaran, beserta data validasi dan evaluasi kinerja aplikasi yang dibuat.

Metode penelitian yang digunakan untuk penelitian ini menggunakan metode *Research and Development* (R&D), yang melibatkan proses sistematis untuk merancang inovasi produk baru dengan mengembangkan atau meningkatkan produk atau layanan yang sudah ada (Gustiani, 2019). Proses ini mengikuti tahapan yang diadaptasi dari metode spiral yang diajukan oleh (B. W. Boehm, 1995) memberikan fleksibilitas dalam proses pengembangan sistem, khususnya dalam hal evaluasi berulang dan perbaikan secara bertahap. Pada Gambar 3.1 ditampilkan gambaran diagram model spiral *original* yang diusulkan (B. Boehm, 2000).



Gambar 3.1 Diagram metode spiral *original* (B. Boehm, 2000)

Model ini menggabungkan unsur-unsur dari pendekatan *waterfall* dan iteratif, yang memungkinkan proses pengembangan dilakukan dalam beberapa tahap iterasi. Diagram tahapan satu iterasi penelitian pendekatan spiral yang akan dilakukan ditampilkan pada Gambar 3.2.



Gambar 3.2 Prosedur penelitian yang akan dilakukan berdasarkan satu iterasi metode Spiral

Setiap siklus pada satu tahap iterasi metode spiral, setiap iterasi mencakup empat kegiatan utama:

1. Penentuan Tujuan

Tahap ini bertujuan untuk mengidentifikasi masalah dan observasi terhadap kebutuhan dan kekurangan awal maupun sistem pada iterasi sebelumnya. Kemudian merumuskan tujuan pada iterasi tahap ini berdasarkan hasil identifikasi dan observasi tersebut.

2. Analisis dan Penyelesaian Resiko

Tahap ini berfokus untuk mengidentifikasi dan mengevaluasi potensi masalah yang dapat menghambat keberhasilan pengembangan sistem. Risiko-risiko tersebut bisa berasal dari berbagai aspek, seperti ketidaksesuaian kebutuhan pengguna, keterbatasan teknologi yang digunakan, kesalahan desain, atau kendala dalam integrasi sistem. Setelah risiko diidentifikasi, dilakukan analisis terhadap dampak dan kemungkinan terjadinya risiko tersebut, lalu dirancang strategi mitigasi untuk meminimalkan atau menghindari dampaknya. Tahap ini penting untuk memastikan

bahwa setiap iterasi pengembangan dapat berjalan lebih efisien dan risiko yang sama tidak terulang di siklus berikutnya.

3. Pengembangan dan Evaluasi

Pada tahap ini, prototipe awal yang telah terbentuk dikembangkan. Kinerja sistem diamati untuk menemukan kesalahan dan kekurangan yang dapat dikembangkan denga penambahan fitur. Pada tahap ini juga, performa sistem deteksi dan rekognisi yang digunakan diperbaiki dengan menyelesaikan masalah-masalah yang timbul pada percobaan. Selanjutnya, aplikasi ini diimplementasikan dan diuji memastikan bahwa aplikasi beroperasi dengan baik.

4. Evaluasi dan Pertimbangan iterasi berikutnya

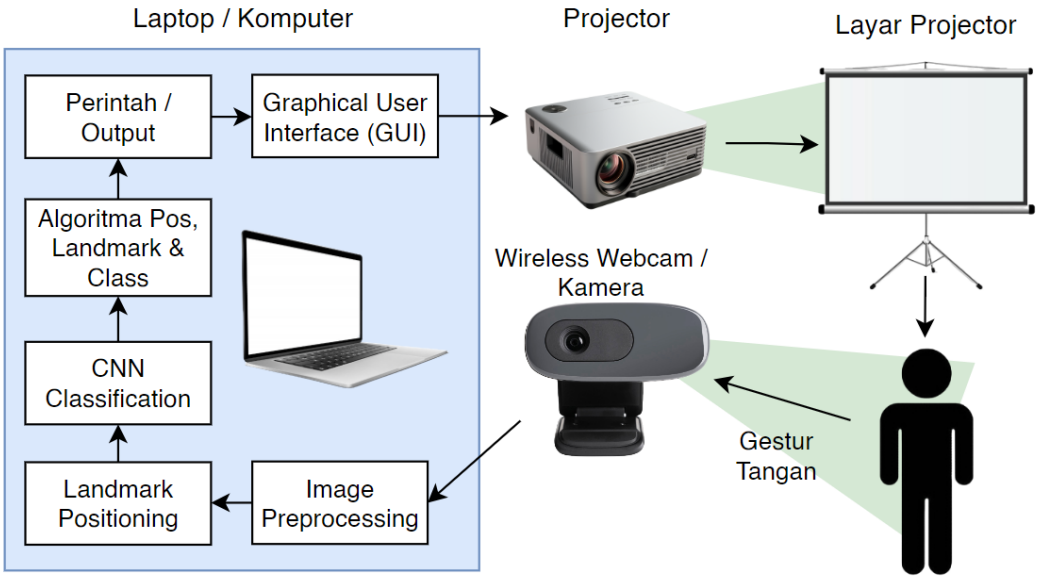
Pada tahap ini, kinerja dan fungsionalitas sistem aplikasi dievaluasi berdasarkan kinerja teknis, efektivitas aplikasi, efisiensi aplikasi, maupun pengukuran performa seperti akurasi, dll. jika diperlukan, sehingga mendapat kesimpulan serta umpan balik mengenai kemudahan penggunaan, dan potensi pengembangan aplikasi selanjutnya.

Dengan menggunakan pendekatan spiral, pengembangan sistem dapat dilakukan secara bertahap dan adaptif, sehingga memudahkan peneliti dalam menyesuaikan kebutuhan pengguna berdasarkan hasil evaluasi tiap siklus. Pendekatan ini sangat sesuai untuk penelitian ini karena melibatkan interaksi pengguna dan membutuhkan penyempurnaan berkelanjutan, baik dari sisi arsitektur, antarmuka, maupun performa sistem.

3.2 Prosedur Perancangan Sistem

Sistem yang dirancang bertujuan untuk menghasilkan pengaplikasian rekognisi gestur tangan yang dapat melakukan: Pertama, mengoperasikan komputer dengan mengontrol kursor, melakukan klik kiri dan kanan, serta memicu tombol perintah yang sering digunakan selama presentasi seperti *next/previous slide*, *escape*, *black/unblack screen*, dll. melalui posisi dan klasifikasi gestur tangan. Kedua mengelola kuis, termasuk membuat, mengedit, menyimpan, menampilkan, dan memulai tes/kuis. Sistem ini akan memilih jawaban pilihan ganda menggunakan gestur tangan, menampilkan progres kuis, serta menunjukkan hasil akhir atau skor setelah kuis selesai.

Untuk membangun project ini, Python dipilih karena kesesuaian dan banyaknya *library* serta *framework* yang tersedia untuk mendukung mendukung pengembangan. Untuk tugas *computer vision*, OpenCV digunakan untuk memproses dan memanipulasi video serta *frame* yang ditangkap oleh sumber kamera. Dalam mendeteksi gestur tangan secara *real-time*, setiap frame video diproses untuk mendeteksi titik *landmark* tangan. Skematik rancangan sistem aplikasi *desktop* secara konvensional diperlihatkan pada Gambar 3.3.



Gambar 3.3 Skematik rancangan pengaplikasian sistem *desktop*

Setelah pengembangan sistem dilakukan pada platform *desktop* untuk diaplikasikan secara lokal, lalu melalui *testing* dan validasi. Aplikasi kemudian dikembangkan lebih lanjut pada iterasi pengembangan selanjutnya sesuai objektif baru. Aplikasi ditujukan agar dapat dijalankan secara daring melalui platform web. Tujuan utama dari penerapan sistem ini pada platform web adalah untuk mendukung pelaksanaan *online proctored exam* dengan gestur tangan untuk mencegah kecurangan yang fleksibel dan dapat diakses dari mana saja selama tersedia koneksi *internet* dan perangkat *laptop* dengan kamera.

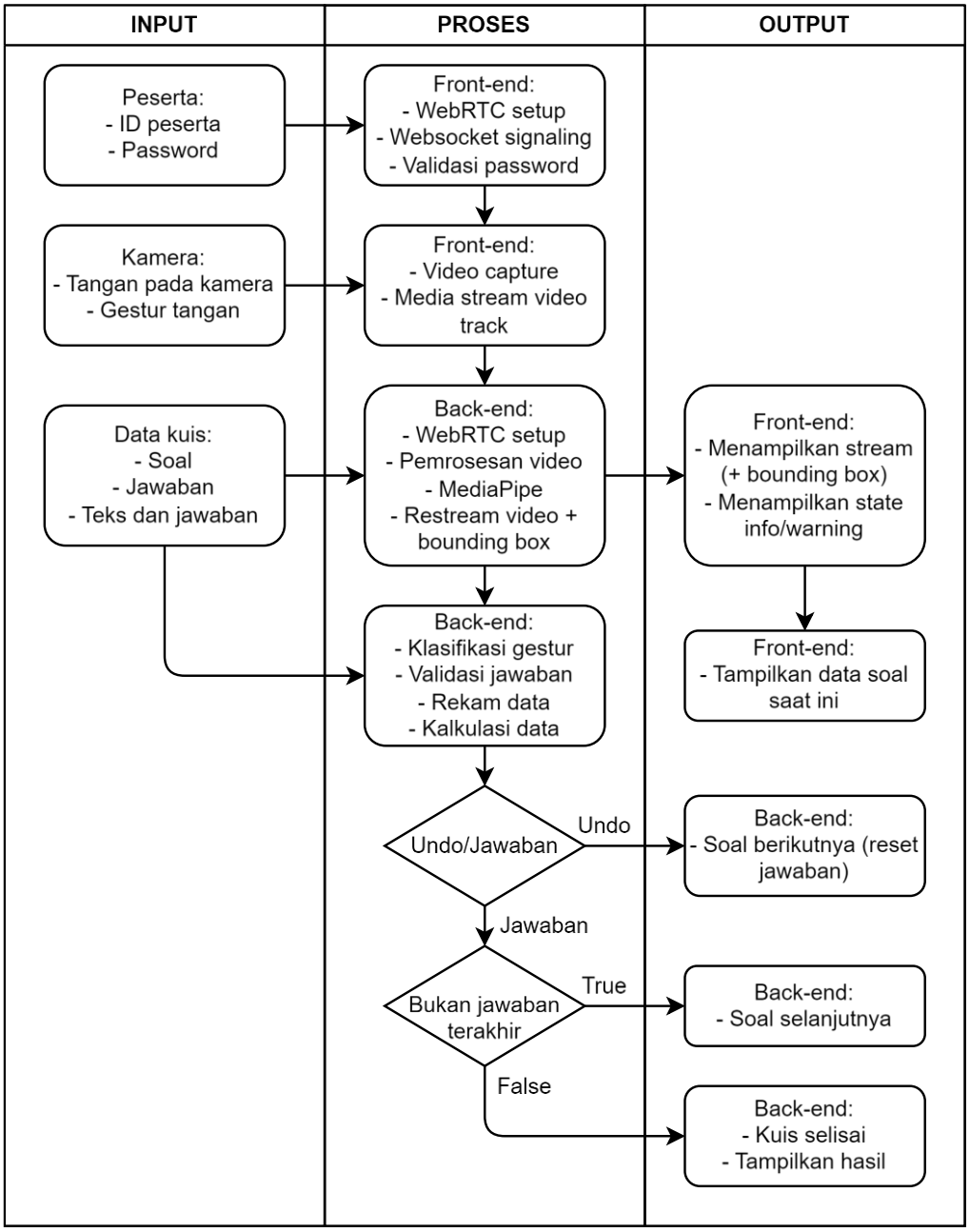
Aplikasi kuis berbasis gestur tangan diadaptasi pada versi web dengan beberapa fitur tambahan yang diperlukan, diantaranya:

- *Timer* dari sisi *server* untuk mematasi waktu pengerjaan tes/kuis yang tidak bisa dikontrol oleh peserta tes.

- Algoritma *monitoring* yang mewajibkan peserta tes menjaga tangan tetap berada dalam area tangkapan kamera selama sesi berlangsung, sehingga mencegah interaksi tidak sah di luar jangkauan pengawasan sistem.
- Bar informasi tambahan saat sistem tidak mendeteksi bahwa salah satu atau kedua tangan keluar dari jangkauan kamera. Agar lebih jelasnya, perhatikan Gambar 3.4.

Gambar 3.4 Gambaran *proctoring* dan pengerjaan tes menggunakan gestur tangan

Sistem web ini mengandalkan teknologi WebRTC untuk melakukan *streaming* video dari browser pengguna ke server *back-end*. Arsitektur ini memungkinkan proses evaluasi berlangsung otomatis dan *real-time*. *Track video webcam* pengguna diproses di sisi *server* menggunakan aiortc untuk deteksi tangan dan klasifikasi gestur secara *real-time*, dengan Django sebagai *framework backend*. Proses *signaling* webRTC dilakukan melalui WebSocket dengan Django *channels* pada bagian *server*, yang mana setelah koneksi WebRTC terhubung, maka koneksi WebSocket tidak lagi dibutuhkan, sehingga diterminasi. Diagram *cross-functional* proses sistem web berdasarkan fungsi *input*, proses dan *output* ditampilkan pada Gambar 3.5.



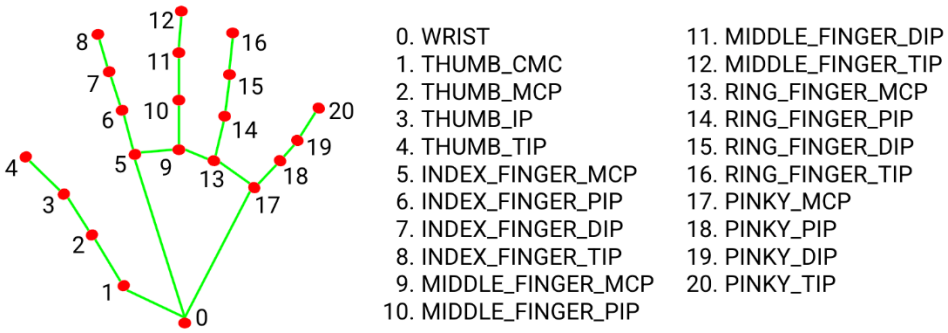
Gambar 3.5 Diagram *cross-functional* aplikasi web

3.2.1 Observasi Teknis dan Pendekatan Solusi Sistem

Pada tahap ini, kegiatan penelitian difokuskan pada pengumpulan informasi serta pemilihan algoritma dan metode yang akan digunakan untuk membangun sistem rekognisi gestur tangan. Tujuannya adalah memastikan solusi yang dipilih mampu mendeteksi, mengenali, dan memproses gestur tangan dengan akurasi tinggi serta dapat diintegrasikan dengan aplikasi pengendali presentasi dan kuis.

Penelitian diawali dengan studi terhadap metode dan algoritma yang tersedia pada beberapa modul dan *framework* Python yang diperlukan dan menyediakan solusi yang dibutuhkan. Metode pada penelitian terdahulu untuk sistem rekognisi gestur tangan juga dijadikan patokan, sehingga mendapat gambaran mengenai metode terbaru dan mendapat ide baru terhadap kekurangan dan masalah yang ada pada metode-metode yang sudah ada. Analisis difokuskan pada:

Teknologi rekognisi tangan: Memahami algoritma dan *framework* yang telah tersedia untuk deteksi tangan, yaitu Mediapipe, modul utama yang digunakan untuk mendeteksi tangan beserta *landmark*-nya. Mediapipe menyediakan model pra-terlatih yang mampu mendeteksi posisi tangan serta 21 titik *landmark* dengan anotasi yang ditampilkan pada Gambar 3.6 secara *real-time*.



Gambar 3.6 *Landmark* tangan MediaPipe (Fernandez, 2023)

Berdasarkan gambar kerangka dan anotasi *landmark* pada Gambar 3.6, dapat diklasifikasikan beberapa gestur yang dibutuhkan berdasarkan perbedaan sudut kerangka dan posisi *landmarknya*. Karena data berupa gambar, dan sudut kerangka, maka CNN menggunakan TensorFlow dan Keras dipilih untuk mengklasifikasi gestur tangan. *Dataset* gambar dikumpulkan menjadi beberapa kelas gestur untuk menjadi data pelatihan model CNN.

Pada tahap pengembangan solusi, penelitian ini mengimplementasikan serangkaian modul yang terintegrasi. Modul pengenalan gestur tangan menggunakan teknologi *computer vision* dibangun untuk mendeteksi dan menginterpretasikan berbagai gestur tangan pengguna. Output dari modul pengenalan gestur tangan ini kemudian dihubungkan dengan pynput, sebuah *library* Python yang memungkinkan kontrol programatik terhadap perangkat input komputer. Pynput dimanfaatkan untuk mentranslasikan gestur tangan yang

terdeteksi menjadi perintah kontrol kursor *mouse* dan tombol *keyboard* seperti panah kiri/kanan untuk navigasi *slide* presentasi, tombol *escape*, dan perintah *keyboard* lainnya sesuai kebutuhan aplikasi.

Antarmuka pengguna (user interface) merupakan komponen krusial dalam aplikasi ini, sehingga *framework* PyQt5 dipilih sebagai teknologi pengembangan GUI. *Qt Designer* dimanfaatkan untuk merancang tampilan visual aplikasi dengan pendekatan *drag-and-drop* yang intuitif, memastikan antarmuka yang terorganisir dan mudah digunakan. Kombinasi PyQt5 dan *Qt Designer* memungkinkan pembuatan antarmuka yang responsif dan estetis, mendukung interaksi pengguna dengan sistem pengenalan gestur tangan dan sistem manajemen kuis yang telah dikembangkan. Pendekatan ini memastikan pengalaman pengguna yang mulus dan intuitif saat menggunakan aplikasi.

Pemilihan arsitektur web dalam pengembangan sistem ini didasarkan pada kebutuhan untuk memproses video pengguna secara *real-time* dari sisi *server*, serta mempertahankan kinerja aplikasi yang ringan dan responsif pada sisi klien. Oleh karena itu, digunakan kombinasi teknologi berikut:

- **React JS** digunakan sebagai *front-end framework* karena mendukung pengembangan antarmuka pengguna yang modular, reaktif, dan cepat. React mempermudah pengelolaan *state* dan komunikasi antar komponen, yang penting dalam konteks aplikasi interaktif berbasis kuis dan video *real-time*.
- **WebRTC** dipilih untuk mengirim *video track* dari pengguna ke *server* tanpa perlu instalasi *plugin* tambahan. WebRTC memungkinkan komunikasi *peer-to-peer* yang efisien, dan dalam kasus ini digunakan dalam mode *client-to-server* melalui media *server* berbasis Python.
- **aiortc** digunakan di sisi backend sebagai *library* Python untuk menerima dan memproses *media stream* dari WebRTC. aiortc dipilih karena mendukung protokol WebRTC secara *native* dalam Python dan memungkinkan integrasi langsung dengan modul *computer vision* yang dikembangkan.
- **WebSocket** digunakan untuk *signaling*, yaitu proses pertukaran *metadata* antara klien dan *server* sebelum koneksi WebRTC dapat dibentuk. WebSocket dipilih

karena menyediakan komunikasi dua arah yang efisien dan *real-time*, cocok untuk kebutuhan *signaling* yang ringan namun cepat.

- **Django + Daphne** digunakan sebagai *server* web berbasis *Asynchronous Server Gateway Interface* (ASGI), yang mendukung koneksi WebSocket secara *native*. Daphne dipilih sebagai *server* karena kompatibel dengan Django dan mendukung protokol *asynchronous* yang dibutuhkan oleh aiortc dan WebSocket secara bersamaan.
- **Ngrok** digunakan selama tahap pengujian untuk membuka akses server lokal ke *internet* dengan membuat *tunnel* aman. Hal ini memudahkan pengujian jarak jauh oleh pengguna tanpa perlu melakukan *deployment* penuh ke *cloud*.

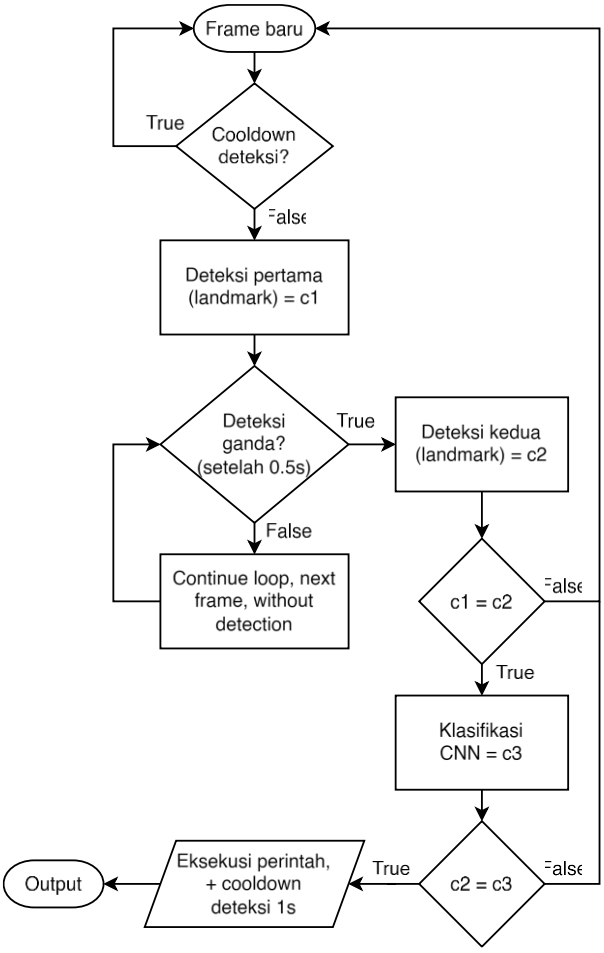
3.2.2 Perancangan Algoritma

Mengandalkan landmark tangan saja untuk mengenali gestur tidak efektif dalam membedakan berbagai macam gestur secara akurat, terutama saat jumlah variasi gestur meningkat. Oleh karena itu, algoritma sistem dirancang lebih lanjut untuk menghasilkan aplikasi yang secara efektif memenuhi kebutuhan dengan menggunakan solusi sistem dari sub-bab sebelumnya. Proses ini terdiri dari beberapa tahap yang saling terintegrasi, yaitu deteksi tangan, ekstraksi fitur, klasifikasi gestur, dan pemetaan aksi/output.

Pemanfaatan posisi landmark dibatasi hanya untuk deteksi sederhana salah satunya mengenali apakah jari dalam posisi terangkat atau yang tidak (terlipat). Selain itu, sistem juga dilengkapi dengan detektor tambahan berbasis posisi, pembanding posisi horizontal antar jari, serta pembanding posisi jari terhadap pergelangan tangan, untuk menilai bentuk dan arah tangan secara lebih menyeluruh (hand dimensionality). Untuk klasifikasi akhir dari gestur tangan, digunakan model CNN yang dilatih.

Dengan menggabungkan dua lapisan deteksi ini, yaitu deteksi posisi awal dan klasifikasi menggunakan CNN, disertai dengan jeda kecil antar proses deteksi, sistem diharapkan mampu mengenali gestur tangan dengan tingkat presisi yang tinggi dan meminimalisir kesalahan saat tangan berada di depan kamera tanpa maksud memberikan isyarat. Proses klasifikasi dilakukan sebagai validasi akhir terhadap gestur yang terdeteksi, sehingga tidak memengaruhi performa aplikasi

secara keseluruhan jika dilakukan pada setiap *frame*, baik dari sisi respons maupun kecepatan *frame*, kecuali pada saat pengambilan keputusan, yang dampaknya dinilai sangat kecil. Secara umum, alur proses algoritma validasi mengambil keputusan dapat dijelaskan dengan sebuah diagram proses yang ditampilkan pada Gambar 3.7.



Gambar 3.7 Diagram proses algoritma validasi dan mengambil keputusan

Metodologi algoritma yang dibangun hingga pemetaan aksi hingga solusi validasi berdasarkan kelemahan dan kekurangan dalam pemetaan aksi, lebih jelasnya sebagai berikut:

1. Deteksi landmark tangan

Sistem menggunakan *MediaPipe Hands* dari Google, sebuah *library* berbasis *machine learning* yang mampu mendeteksi tangan dan 21 titik *landmark* secara *real-time* pada *frame* video. Titik-titik ini mencakup ujung dan sendi setiap jari serta pergelangan tangan.

2. Ekstraksi fitur sudut

Daripada menggunakan koordinat mentah dari titik *landmark*, sistem menghitung sudut antar tiga titik yang membentuk setiap sendi jari. Representasi sudut ini memiliki sifat invarian terhadap translasi, rotasi, dan skala, sehingga lebih stabil dibandingkan penggunaan koordinat *landmark* secara langsung. Selain itu, sudut juga mengekstraksi fitur menjadi setidaknya sepertiga jumlah fitur sebelumnya, yang membuat beban komputasi *real-time* lebih ringan dibanding posisi *landmark* yang memiliki 3 nilai (x, y, z) setiap *landmark*-nya.

Diberikan *landmark* tangan $L = \{l_0, l_1, \dots, l_{20}\}$ dimana setiap *landmark* $l_i = (x_i, y_i, z_i)$. Maka untuk tiga *landmark* berurutan yang membentuk sudut sendi (misal *landmark* l_a, l_b, l_c dimana l_b adalah titik *vertex* sendi), dihitung dua vektor arah:

$$\overrightarrow{v_1} = l_a - l_b \text{ dan } \overrightarrow{v_2} = l_c - l_b \tag{3.1}$$

Langkah selanjutnya menghitung sudut menggunakan *dot product*:

$$\theta = \cos^{-1}\left(\frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\|_2 \cdot \|\overrightarrow{v_2}\|_2}\right) \tag{3.2}$$

Untuk setiap jari $f \in \{ibu\ jari, telunjuk, tengah, manis, kelingking\}$ yang dihasilkan MediaPipe:

- $F_{ibu\ jari} = [0, 1, 2, 3, 4]$
- $F_{telunjuk} = [0, 5, 6, 7, 8]$
- $F_{tengah} = [0, 9, 10, 11, 12]$
- $F_{manis} = [0, 13, 14, 15, 16]$
- $F_{kelingking} = [0, 17, 18, 19, 20]$

Untuk jari f dengan urutan *landmark* $F_f = [i_0, i_1, i_2, i_3, i_4]$, sudut yang dihitung dari *landmark* jari tersebut adalah $\Theta_f = \{\theta_j : j = 1, 2, 3\}$. Berdasarkan persamaan (3.2), sudut dapat dihitung secara langsung terhadap urutan *landmark*:

$$\theta_j = \cos^{-1}\left(\frac{(l_{i_{j-1}} - l_{i_j}) \cdot (l_{i_{j+1}} - l_{i_j})}{\left\|l_{i_{j-1}} - l_{i_j}\right\|_2 \cdot \left\|l_{i_{j+1}} - l_{i_j}\right\|_2}\right) \tag{3.3}$$

Seluruh sudut yang dihitung diubah ke dalam bentuk numpy *array* A:

$$\mathrm{A} = [\Theta_{ibu\ jari}, \Theta_{telunjuk}, \Theta_{tengah}, \Theta_{manis}, \Theta_{kelingking}]^T$$

$$\mathrm{A} \in \mathbb{R}^{15}$$

Selain 15 fitur sudut sendi jari tersebut, penelitian ini menambahkan 3 fitur sudut tambahan yang dihitung berdasarkan hubungan geometris antara ujung jari. Penambahan fitur ini bertujuan untuk meningkatkan kemampuan model dalam mengenali gestur yang melibatkan konfigurasi spesifik antar ujung jari untuk mengenali varians antar jari, tidak hanya sendi setiap individu jari. Tiga sudut antar ujung jari yang diekstraksi:

- ibu jari-telunjuk-tengah (telunjuk sebagai vertex)
- telunjuk-tengah-manis (tengah sebagai vertex)
- tegah-manis-kelingking (manis sebagai vertex)

Untuk menghitung sudut-sudut tersebut digunakan rumus yang sama dengan persamaan untuk menghitung sudut sendi jari, yaitu menggunakan persamaan (3.3). Sehingga menghasilkan total fitur sudut menjadi 18, yang lebih efisien dibandingkan menggunakan 63 koordinat *landmark* mentah.

3. Klasifikasi Gestur (Inference)

Array sudut yang dihasilkan dari proses ekstraksi dimasukkan ke dalam model CNN untuk diklasifikasikan. *Output* dari CNN berupa label kelas gestur tangan. Arsitektur dan proses *training*, serta model final akan dijelaskan pada Bab 4 Hasil dan Pembahasan. Kompleksitas komputasi berdasarkan model yang dilatih dengan bentuk fitur yang telah dijelaskan:

- Kompleksitas Waktu: $O(n)$ dimana n adalah jumlah sudut ($n = 18$)
- Kompleksitas Ruang: $O(n)$ untuk menyimpan fitur sudut

4. Pemetaan ke Aksi Sistem

Setiap gestur yang dikenali akan dipetakan ke fungsi sesuai dengan kelas pada model CNN. Namun, sistem perlu memetakan hasil klasifikasi tersebut ke dalam aksi-aksi tertentu yang relevan, seperti navigasi slide dalam presentasi atau pemilihan opsi dalam kuis. Namun, dalam praktiknya, terdapat beberapa tantangan dan potensi kesalahan dalam interpretasi gestur yang perlu ditangani melalui logika

validasi tambahan. Beberapa kelemahan yang ditemukan dalam proses pemetaan aksi sistem, beserta solusinya antara lain:

A. *Smoothening* pergerakan kursor

Pada fitur kontrol presentasi, sistem menyediakan fungsi pengendalian kursor berdasarkan pergerakan posisi tangan. Namun, jika posisi tangan berpindah terlalu cepat, kursor dapat bergerak tidak stabil atau *flickering*. Untuk mengatasi hal ini, diterapkan proses *smoothening* dengan metode interpolasi posisi kursor berdasarkan rata-rata pergerakan dalam beberapa *frame* terakhir. Maka kecepatan kursor dapat diperhalus, dengan perhitungan kecepatan pergerakan kursor:

$$V = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{\Delta t} \tag{3.4}$$

Kecepatan diperhalus dengan konstanta S:

$$(x, y)_n = (x, y)_{n-1} + \frac{((x, y)_i - (x, y))_{n-1}}{S} \tag{3.5}$$

Maka hasilnya, pergerakan kursor:

- Semakin tinggi kecepatan gerak tangan, sistem otomatis mengurangi sensitivitas perpindahan posisi kursor untuk menjaga kelancaran visual.
- Pendekatan ini menghasilkan pergerakan kursor yang lebih halus dan natural, tanpa mengurangi responsivitas (derivatif terhadap V).

B. Kesalahan Akibat Gestur yang Tidak Disengaja

Gestur tangan yang berubah-ubah secara cepat atau tidak disengaja dapat terdeteksi sebagai *input* valid dan memicu aksi yang tidak diinginkan. Untuk mengurangi hal ini, sistem dilengkapi dengan mekanisme validasi waktu berupa deteksi ganda. Gestur dianggap valid hanya jika tetap terdeteksi konsisten dalam periode waktu tertentu. Pendekatan ini membantu menyaring perubahan bentuk tangan yang bersifat transisional atau tidak disengaja.

C. Arah dan Kedudukan Tangan Tidak Konsisten

Karena model CNN dilatih berdasarkan sudut antar *landmark*, ia bersifat invarian terhadap rotasi global tangan. Hal ini memungkinkan orientasi tangan yang tidak menghadap kamera tetap dikenali, yang bisa menimbulkan ambiguitas. Untuk

mengatasi hal ini, sistem menambahkan filter orientasi tangan berdasarkan posisi relatif *landmark*:

- Arah telapak tangan: jika tangan kanan, maka *landmark* ibu jari harus berada di sisi kiri *landmark* buku jari (sebaliknya terhadap tangan kiri).
- Orientasi vertikal tangan: posisi relatif secara vertikal antara *landmark* pergelangan tangan dengan *landmark* buku jari.
- Keadaan kelima jari (terlipat atau terbuka)

Perhitungan dilakukan dengan membandingkan posisi *landmark* tangan dengan konfigurasi orientasi tangan gestur, hal ini dilakukan sebelum klasifikasi CNN, lihat Gambar. Jika susunan *landmark* terbalik (tangan tidak menghadap kamera), maka gestur dinyatakan tidak valid dan tidak diproses lebih lanjut.

D. Pengendalian Aksi Hanya Berdasarkan Gestur yang Stabil

Aksi sistem seperti "*next slide*", "jawaban pilihan ke-3", atau "aktifkan mode kursor" hanya akan dijalankan jika gestur valid terdeteksi secara konsisten. Ini menghindari kondisi di mana satu gestur diklasifikasi benar tetapi dilakukan terlalu cepat atau dalam kondisi posisi tangan tidak stabil. Dengan kombinasi validasi orientasi dan durasi, sistem menjadi lebih tahan terhadap kesalahan deteksi dan gerakan tangan yang tidak disengaja.

3.2.3 Pengembangan Model

Tahap pengumpulan dan *labeling dataset* dilakukan untuk menyediakan data yang diperlukan dalam proses pelatihan model klasifikasi CNN yang akan digunakan. *Dataset* yang dikumpulkan melalui program khusus untuk mengkoleksi data dari *frame* tangkapan kamera. gestur tangan dalam berbagai posisi, orientasi, dan kondisi pencahayaan untuk memastikan kemampuan sistem dalam mengenali gestur secara akurat. Daftar gestur dibagi kedalam 3 tabel terpisah, dimana daftar gestur kecuali pada gestur kursor akan dijadikan gestur dataset model. Daftar gestur yang digunakan beserta perintahnya ditampilkan pada Tabel 3.1, Tabel 3.2, dan Tabel 3.3.

Tabel 3.1 Gestur tangan mode *'cursor'* dan perintahnya (tangan kanan, sudut pandang kamera, *mirrored*)

Gestur	Perintah
	Menggerakkan kursor, mengikuti posisi ujung jari telunjuk
	Tahan posisi kursor, (ibu jari dilipat)
	Klik kiri, menurunkan jari telunjuk sesaat (imitasi gerakan klik kiri menggunakan *mouse*)
	Klik kanan, menurunkan jari tengah sesaat (imitasi gerakan klik kanan menggunakan *mouse*)
	Aktifkan mode *keys*

Tabel 3.2 Gestur tangan mode *'key'* dan perintahnya (tangan kanan, sudut pandang kamera, *mirrored*)

Gestur	Perintah
	'esc' *Key* - *Escape* atau keluar
	*'right arrow' Key* – Kanan atau *next slide*
	*'left arrow' Key* – Kiri atau *previous slide*

	*'b' Key* – Hitamkan layar *on/off*
	Aktifkan mode kursor

Tabel 3.3 Gestur tangan fitur kuis dan perintahnya (tangan kanan, sudut pandang kamera, *mirrored*)

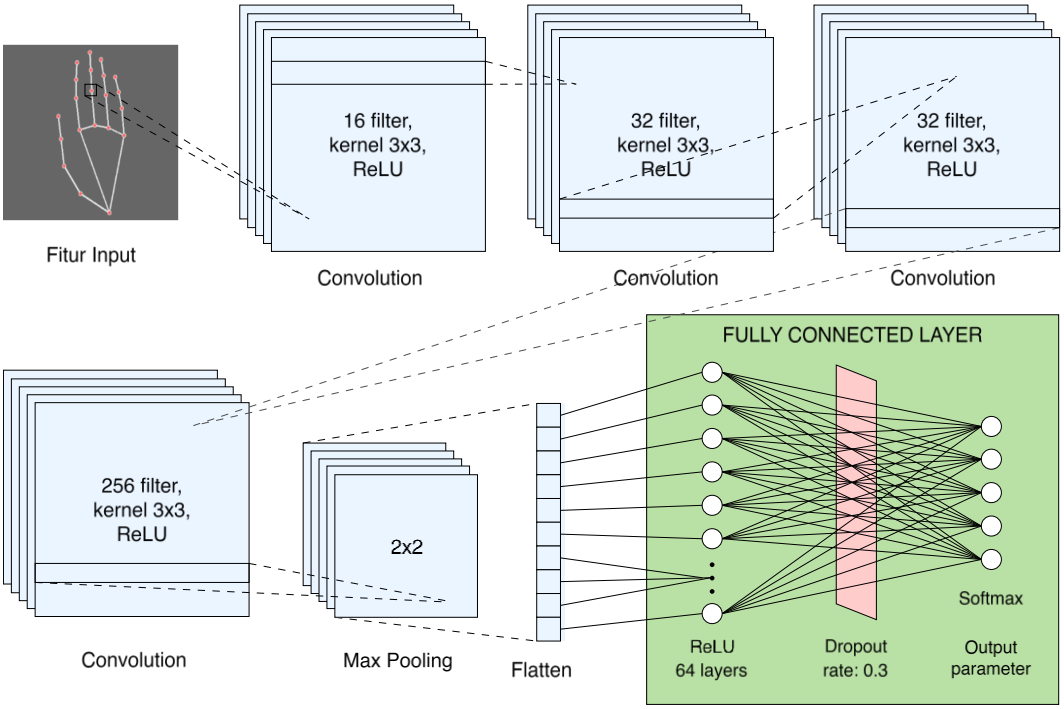
Gestur	Perintah
	*Undo*
	Jawaban 1
	Jawaban 2
	Jawaban 3
	Jawaban 4

Pengambilan gambar tangan dari kamera secara langsung menggunakan kode khusus yang dirancang untuk menangkap frame video dan menyimpan data secara cepat yang dilakukan dengan peralatan dan lingkungan yang dibutuhkan, serta format *dataset* sebagai berikut:

- **Peralatan**: *Webcam* beresolusi tinggi dengan *frame rate* minimal 30 *frame per second* (FPS) untuk menangkap fitur gestur secara akurat.

- **Lingkungan**: Proses pengambilan data dilakukan di ruangan dengan pencahayaan tinggi dan rendah normal di siang hari (lampu menyala dan mati) untuk memastikan dataset mencakup berbagai kondisi yang memungkinkan.
- **Gambar**: Frame video dalam format gambar warna *red-green-blue* (RGB), disimpan dalam format .png.
- **Anotasi gestur**: Fitur sudut setiap kelas gestur tangan yang dihasilkan, berbentuk numpy *array*, disimpan dalam format .npy.
- **Label**: Klasifikasi gestur berdasarkan jenis interaksi, dikumpulkan dalam folder berbeda untuk gambar, masing masing file .npy dengan nama yang berbeda untuk fitur sudut.

Arsitektur CNN berdasarkan ukuran data input dibangun dengan komposisi *hyperparameter* yang disesuaikan berdasarkan performa *training* dan validasi awal yang menggunakan *epoch* maksimal dengan *early stopping*. Representasi visual dari arsitektur inisiasi model CNN pada iterasi awal ditampilkan pada Gambar 3.8.



Gambar 3.8 Arsitektur model CNN inisiasi/awal

3.2.4 Deployment

Proses *deployment* aplikasi desktop dilakukan dengan membangun antarmuka pengguna menggunakan PyQt5 dan Qt Designer, yang kemudian

dikompilasi menjadi aplikasi *desktop*. Pemisahan antara desain UI dan kode logika membantu pengembangan lebih terstruktur dan fleksibel. Aplikasi desktop dirancang untuk secara otomatis mendeteksi dan menggunakan indeks kamera tertinggi yang tersedia pada sistem, yang umumnya merujuk pada *webcam* eksternal. Pendekatan ini digunakan karena skenario penggunaan utama aplikasi desktop ini ditujukan untuk lingkungan kelas atau presentasi, di mana *webcam* eksternal digunakan bersama dengan proyektor. Hal ini memungkinkan pengguna untuk mengontrol fitur presentasi maupun kuis secara interaktif melalui gestur tangan tanpa perlu menulis kode atau melalui IDE.

Untuk mengimplementasikan fungsionalitas tes/kuis, dibangun sistem manajemen kuis yang komprehensif. Pada aplikasi *desktop* dilengkapi dengan kemampuan untuk membuat, mengedit, menghapus, dan menyimpan soal kuis dalam format CSV. Fitur pendukung ditambahkan untuk memungkinkan integrasi gambar pada soal kuis, sehingga memperkaya pengalaman dan variasi soal yang dapat disajikan. Seluruh data kuis disimpan secara terstruktur sehingga dapat diakses dan dikelola dengan efisien oleh aplikasi saat mode permainan kuis dijalankan.

Pada aplikasi web, proses komunikasi video *real-time* antara pengguna dan *server back-end* dalam sistem ini terdiri atas beberapa tahap utama yang melibatkan akuisisi video dari kamera, negosiasi koneksi *peer-to-peer*, transmisi *stream* video, pemrosesan *computer vision*, dan pengiriman hasil ke pengguna. Aplikasi web dalam penelitian ini dikembangkan menggunakan React JS sebagai antarmuka pengguna (front-end) dan Django sebagai pengelola logika *back-end* serta pemrosesan *computer vision* secara *real-time*.

Komunikasi video antara klien dan *server* dilakukan menggunakan protokol WebRTC, di mana *media stream track* video dari kamera pengguna dikirim ke server untuk melalui proses *computer vision*. Berikut adalah penjabaran setiap tahapnya:

1. Akuisisi dan inisialisasi media di *front-end*

Proses dimulai dari sisi *front-end* dengan pemanggilan fungsi getUserMedia() untuk mengakses kamera. Fungsi ini akan menghasilkan objek MediaStream yang

berisi *track* video dari kamera pengguna. Stream ini ditampilkan di antarmuka React agar pengguna dapat melihat pratinjau secara langsung. Track video dari stream kemudian ditambahkan ke objek RTCPeerConnection menggunakan metode addTrack(), yang memungkinkan WebRTC mengirimkan *stream* tersebut ke *server*.

2. Proses *signaling* menggunakan WebSocket

Agar koneksi *peer-to-peer* dapat terhubung, perlu ada pertukaran *metadata* SDP dan *ICE Candidate*. Pertukaran ini dilakukan menggunakan WebSocket sebagai saluran *signaling* yang mencakup:

- *Client* membuat *offer* (createOffer()), mengatur sebagai *local description* (setLocalDescription()), dan mengirim ke *server* melalui WebSocket.
- Server (aiortc) menerima *offer*, membuat *answer* (createAnswer()), mengatur sebagai *local description*, dan mengembalikannya ke *client*.
- Keduanya bertukar *ICE Candidate* untuk menentukan jalur komunikasi terbaik.

3. Penerimaan media stream di *back-end* (Django + aiortc)

Server Django yang menjalankan modul aiortc menerima koneksi dan menyusun objek RTCPeerConnection. *Track* video yang diterima akan didekode menjadi *frame-frame* video menggunakan fungsi internal aiortc VideoStreamTrack.

4. Pemrosesan *computer vision*

Setiap *frame* yang diterima diproses secara *real-time* dalam *server*.

5. Pengembalian hasil ke *front-end*

Hasil video dari pemrosesan yang memiliki *bounding box*, beserta data keputusan dan lainnya mengenai tes yang dipengaruhi oleh kamera dikirim kembali menggunakan *video track stream* dari aiortc *server* ke webrtc pada *front-end*.

6. Penutupan Koneksi

Setelah sesi kuis selesai, koneksi WebRTC ditutup. Proses ini mencakup penghentian *track* video, penghapusan *peer connection*, dan pembersihan *resource back-end*.

3.3 Prosedur Pengujian dan Evaluasi

Untuk memastikan bahwa sistem yang dikembangkan dapat berjalan sesuai dengan kebutuhan pengguna dan spesifikasi yang telah dirancang, dilakukan serangkaian prosedur pengujian dan evaluasi. Pengujian ini mencakup pengujian fungsionalitas sistem, performa model klasifikasi gestur berbasis CNN, serta evaluasi dari calon pengguna terhadap aspek kegunaan dan efektivitas sistem. Berikut adalah uraian dari setiap metode pengujian dan evaluasi yang digunakan:

A. Pengujian fungsionalitas sistem (black-box testing)

Pengujian fungsional dilakukan menggunakan metode *black-box testing* untuk memeriksa apakah setiap fitur pada aplikasi web berfungsi sesuai harapan. Teknik yang digunakan adalah *equivalence partitioning*, di mana setiap *input* yang mungkin dikelompokkan ke dalam kelas-kelas yang ekuivalen untuk mengidentifikasi kesalahan atau kegagalan sistem. Fokus pengujian dilakukan pada sistem hasil akhir dari iterasi terakhir dalam model pengembangan spiral. Pengujian ini dilakukan tanpa melihat struktur *internal* kode program, melainkan berdasarkan interaksi pengguna terhadap antarmuka dan hasil keluaran sistem.

B. Pengujian model CNN dan performa sistem

Model klasifikasi gestur tangan yang digunakan dalam sistem dibangun menggunakan arsitektur CNN 1D dengan dataset fitur sudut gestur tangan. Pengujian terhadap model ini mencakup evaluasi akurasi, presisi, *recall*, dan *F1-score* berdasarkan data uji yang telah dipisahkan pada saat pelatihan. Selain itu, dilakukan pengukuran terhadap performa sistem secara *runtime*, seperti kecepatan respons terhadap input gestur, keterlambatan pemrosesan (latensi), dan kestabilan deteksi selama penggunaan aplikasi, khususnya pada platform web.

C. Evaluasi Pengguna melalui Kuisioner

Evaluasi subjektif dilakukan untuk mengetahui persepsi pengguna terhadap sistem yang telah dikembangkan. Evaluasi ini dilaksanakan melalui penyebaran formulir kuisioner kepada responden yang terdiri dari 30 mahasiswa Program Studi Pendidikan Teknik Otomasi Industri dan Robotika (PTOIR) angkatan 2024 yang sedang menempuh semester kedua pada mata kuliah Dasar Komputer dan

Pemrograman. Kuisioner dirancang berdasarkan beberapa indikator utama dalam evaluasi perangkat lunak, yaitu:

- Usabilitas: sejauh mana sistem mudah dipelajari dan digunakan.
- Efektivitas: ketepatan sistem dalam mendukung tujuan atau fungsinya.
- Reliabilitas: seberapa reliabelsistem untuk digunakan atau direkomendasikan.
- Fungsionalitas: kelengkapan fitur dan kemampuan sistem dalam memenuhi kebutuhan dan tujuan.
- Masukan/Saran: tanggapan atau usulan perbaikan terhadap sistem.

Hasil dari pengisian kuesioner ini akan digunakan sebagai dasar dalam menilai sejauh mana sistem telah memenuhi ekspektasi pengguna dan memberikan arah perbaikan untuk pengembangan selanjutnya.

3.4 Lingkungan Komputasi

Lingkungan komputasi memiliki pengaruh yang sangat penting pada pengembangan teknologi. Pada penelitian ini, lingkungan komputasi menentukan performa dan kecepatan pengembangan sistem computer vision, yang mana melibatkan pelatihan model *machine learning*, sehingga diperlukan lingkungan, perangkat keras, dan perangkat lunak yang memadai. Persyaratan perangkat keras yang digunakan dalam pengembangan pada penelitian ini terdiri dari:

Perangkat yang digunakan yaitu *laptop* pribadi HP Zbook 15 G2 dengan spesifikasi:

- CPU *Intel i7-4810MQ @ 2.80GHz*
- RAM 8,0 GB
- SSD 256 GB
- GPU 0 *Intel HD Graphics 4600*
- GPU 1 *NVIDIA Quadro K2100M (2GB GDDR5)*

Cloud computing dilakukan menggunakan Google Colab dengan spesifikasi:

- CPU *Intel Xeon @2.20 GHz (GPU) @2.30 GHz (TPU)*
- RAM 13 GB
- *Tesla K80 accelerator, 12 GB GDDR5 VRAM, 180 teraflops TPU*
- *Runtime* 12 jam

Pada percobaan dan implementasi, video diambil menggunakan *Webcam* nirkabel HOSODO HSD-P100 dengan spesifikasi:

- Resolusi 2160×1080 *pixel*
- *Aspect ratio* 2:1
- *Frame rate* 30 *FPS*
- *FOV* 88˚

Pada Tabel 3.4 ditunjukkan daftar perangkat lunak (software) yang digunakan dalam penelitian ini.

Tabel 3.4 *Software* yang digunakan

Kategori	Perangkat Lunak
Sistem Operasi	**Windows 11 Pro**, versi 22H2, sistem operasi utama untuk perancangan dan pengembangan.
Alat Pengembangan dan IDE	**VSCode**, versi 1.93.1, sebagai IDE utama untuk perancangan dan pengembangan.
	Google Colab, untuk pengembangan dan pelatihan model dengan akses ke GPU/TPU berbasis *cloud*.
	Jupyter Notebook, untuk eksperimen interaktif dan dokumentasi *source code*.
Manajemen Data dan Kolaborasi	**GitHub**, sebagai tempat dokumentasi dan *project repository.*
	Google Drive, tempat penyimpanan dan dokumentasi.
Bahasa Pemrograman dan *Framework*	**Python**, versi 3.11.4, sebagai bahasa pemrograman utama pada penelitian ini.
	JavaScript, v20.13.1, sebagai bahasa pemrograman *front-end*.
	TensorFlow, versi 2.16.1, sebagai *framework* utama dalam pelatihan model CNN.
	PyQt5, versi 5.15.10, sebagai *framework* pembuatan GUI aplikasi *desktop* python.
	Django, versi 5.1.7, sebagai *framework back-end.*
	React JS, versi 19.0.0, sebagai *framework front-end*.

Tools Lainnya	**OpenCV**, versi 4.10.0.82, modul untuk pekerjaan *computer vision* dan pemrosesan gambar.
	MediaPipe, versi 0.10.14, *framework pipeline machine learning* untuk mendeteksi tangan beserta *landmark*-nya.
	WebRTC, protokol JavaScript untuk komunikasi data video secara *real-time*.
	aiortc, versi 1.13.0, modul python untuk men-*deploy* webrtc pada *back-end* python .
	Daphne, versi 4.1.2, server ASGI python untuk men-*deploy* aplikasi Django.
	Ngrok, layanan *tunneling proxy server* dari *localhost* ke jaringan *internet*.
	draw.io, versi 24.7.5.0, tools untuk membuat diagram dan ilustrasi teknis.