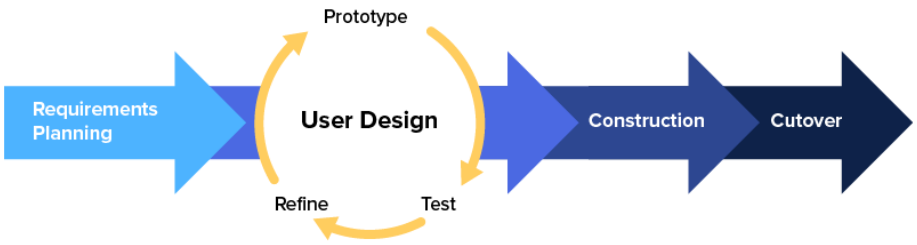
BAB III
METODOLOGI

3.1 Metode Pengembangan Sistem

Penelitian ini menggunakan metode *Rapid Application Development* (RAD) sebagai pendekatan pengembangan sistem. RAD merupakan metode pengembangan yang menekankan pada kecepatan proses pembangunan aplikasi melalui pendekatan umpan ballik, keterlibatan aktif pengguna, serta penggunaan *prototyping* secara berkelanjutan. Metode ini dirancang untuk menghasilkan sistem dalam waktu relatif singkat dengan tetap memperhatikan kualitas dan kesesuaian kebutuhan pengguna. RAD memungkinkan adanya komunikasi yang intens antara pengembang dan pengguna, sehingga setiap perubahan kebutuhan dapat segera diimplementasikan selama proses pengembangan berlangsung (Simanungkalit et al., 2023).

Alasan pemilihan metode RAD dalam penelitian ini didasarkan pada beberapa pertimbangan. Pertama, ruang lingkup penelitian yang terfokus yaitu klasifikasi penyakit pada daun tomat dengan jumlah kelas yang terbatas (empat kelas), sehingga memungkinkan pengembangan dilakukan secara cepat. Kedua, adanya kebutuhan keterlibatan petani sebagai pengguna secara langsung dalam tahap perancangan agar antarmuka aplikasi agar sesuai dengan kebutuhan di lapangan. Ketiga, metode RAD memiliki fleksibilitas terhadap perubahan kebutuhan yang mungkin muncul selama proses pengembangan, terutama pada aspek arsitektur model CNN dan tampilan aplikasi Android.

Secara teoritis, metode RAD terdiri dari empat tahapan utama yang saling terintegrasi Tahapan metode RAD tersebut ditunjukkan pada gambar berikut.



Gambar 3. 1 Tahapan Metode RAD (Simanungkalit et al., 2023)

Berikut adalah penjelasan masing-masing tahapan metode RAD yang diterapkan dalam penelitian ini:

1. *Requirements Planning* (Perencanaan Kebutuhan)

Pada tahap ini, dilakukan wawancara dan observasi langsung kepada petani di Desa Babakanmulya, Kecamatan Cigugur, Kabupaten Kuningan untuk mendefinisikan ruang lingkup sistem, kebutuhan pengguna serta batasan penelitian. Hasil dari tahap ini mencakup identifikasi empat kelas klasifikasi (*Leaf Spot, Leaf Blight, Powdery Mildew*, dan Daun Sehat), spesifikasi *dataset* yang dibutuhkan, serta kebutuhan platform berbasis Android. Proses ini sangat penting untuk memastikan bahwa sistem yang dikembangkan sesuai dengan kebutuhan pengguna dan dapat menyelesaikan permasalahan yang dihadapi.

2. *User Design* (Perancangan Pengguna)

Pada tahap ini dilakukan perancangan sistem dengan melibatkan pengguna secara langsung. Pengembang membuat desain awal berupa prototipe antarmuka aplikasi, model sistem, serta diagram UML yang meliputi Use Case Diagram, Activity Diagram, *Class* Diagram, dan Sequence Diagram. Selain itu, pada tahap ini juga dirancang dengan arsitektur model MobileNetV2 serta alur penyelesaian masalah mulai dari *preprocessing* citra hingga *output* klasifikasi. Tahap *preprocessing* merupakan langkah awal yang dilakukan sebelum citra daun tomat dimasukkan ke dalam model CNN untuk proses pelatihan. Tujuan utama dari tahap ini adalah memastikan bahwa seluruh data citra memiliki format, ukuran, dan nilai piksel yang seragam sehingga model dapat belajar secara optimal. Tanpa *preprocessing* yang tepat, perbedaan ukuran, pencahayaan, dan orientasi antar citra dapat menyebabkan model mengalami kesulitan dalam mengekstraksi fitur yang konsisten, sehingga berdampak pada penurunan akurasi klasifikasi.

Dalam penelitian ini, tahap preprocessing mencakup serangkaian proses sebagai berikut: (1) Resize (pengubahan ukuran), yaitu seluruh citra diubah menjadi resolusi 224×224 piksel sesuai dengan ukuran input standar arsitektur MobileNetV2; (2) Normalisasi piksel, yaitu nilai piksel pada setiap kanal warna

dinormalisasi dari rentang 0–255 menjadi 0–1 dengan cara membagi seluruh nilai piksel dengan angka 255 (rescale = 1./255), sehingga model lebih mudah dalam mempelajari pola pada citra; (3) Augmentasi data, yaitu penerapan teknik transformasi geometris dan fotometris secara acak selama pelatihan, yang meliputi horizontal flip, rotasi hingga 30 derajat, zoom sebesar 20%, pergeseran horizontal dan vertikal masing-masing sebesar 10%, serta transformasi shear hingga 20%. Augmentasi ini bertujuan memperbesar variasi data latih sehingga model lebih mampu menggeneralisasi terhadap kondisi nyata di lapangan. Pengguna kemudian memberikan umpan balik terhadap desain tersebut, sehingga terjadi proses evaluasi dan penyempurnaan secara berulang hingga diperoleh rancangan yang sesuai dengan kebutuhan operasional.

3. *Construction* (Konstruksi)

Tahap konstruksi merupakan fase pengembangan sistem secara nyata berdasarkan desain yang telah disepakati. Pada penelitian ini, tahap konstruksi mencakup dua proses utama yang berjalan secara paralel, yaitu: pembangunan dan pelatihan model *deep learning* menggunakan algoritma CNN dengan arsitektur MobileNetV2 di Google Colaboratory berbasis Python dan TensorFlow, serta pengembangan aplikasi Android menggunakan bahasa pemrograman Kotlin dengan mengintegrasikan model TensorFlow Lite hasil pelatihan. Proses pelatihan model dilakukan menggunakan *dataset* sebanyak 4.000 citra daun tomat dengan pembagian 80% data latih, 10% data validasi, dan 10% data uji. Setelah proses *preprocessing* selesai, tahap selanjutnya sebelum implementasi model adalah validasi model. Validasi model merupakan proses evaluasi terhadap performa model klasifikasi CNN berbasis MobileNetV2 menggunakan data validasi yang telah dipisahkan sejak awal dan tidak digunakan dalam proses pelatihan. Tujuan dari tahap ini adalah memantau kemampuan generalisasi model terhadap data baru selama proses *training* berlangsung, sehingga dapat mendeteksi lebih awal apakah model mengalami *overfitting* maupun *underfitting*.

Dalam penelitian ini, data yang digunakan untuk validasi berjumlah 400 citra (10% dari total dataset), dengan 100 citra per kelas. Performa model selama

validasi dipantau melalui grafik val_accuracy dan val_loss yang dihasilkan pada setiap epoch. Model terbaik yang diperoleh dari proses validasi kemudian disimpan menggunakan mekanisme ModelCheckpoint berdasarkan nilai val_accuracy tertinggi, sebelum selanjutnya dikonversi ke format TensorFlow Lite untuk diimplementasikan ke dalam aplikasi Android. Selain itu, dilakukan juga pengujian secara bertahap untuk memastikan bahwa setiap fungsi sistem berjalan dengan baik.

4. *Cutover* (Implementasi)

Tahap terakhir adalah implementasi sistem ke pengguna. Pada fase ini dilakukan pengujian akhir aplikasi Android menggunakan metode *Black Box Testing* untuk mengetahui fungsi sistem secara menyeluruh. Selain itu, dilakukan pula evaluasi performa model klasifikasi menggunakan *Confusion Matrix* beserta metrik turunannya meliputi *Accuracy, Precision, Recall*, dan *F1-Score*. Sistem yang telah dikembangkan kemudian dijalankan secara penuh dan dievaluasi untuk memastikan bahwa aplikasi telah memenuhi kebutuhan petani dalam mengidentifikasi penyakit daun tomat secara cepat, akurat, dan mudah digunakan melalui perangkat Android.

3.2 Metode Pengumpulan Data

Metode pengumpulan data merupakan teknik yang digunakan oleh peneliti untuk memperoleh informasi yang diperlukan dalam proses penelitian. Data yang diperoleh digunakan sebagai dasar dalam analisis sistem serta perancangan aplikasi klasifikasi penyakit pada daun tomat berbasis Android. Dalam penelitian ini, metode pengumpulan data yang digunakan meliputi observasi, wawancara, kuesioner, dan studi pustaka.

3.2.1 Observasi

Observasi dilakukan dalam dua tahap, yaitu observasi pendahuluan dan observasi menggunakan aplikasi yang telah dikembangkan. Observasi pendahuluan dilakukan dengan pengamatan langsung terhadap kondisi lahan

pertanian tomat di Desa Babakanmulya, Kecamatan Cigugur, Kabupaten Kuningan pada tanggal 30 Januari 2026. Berdasarkan hasil observasi, diketahui bahwa petani masih melakukan identifikasi penyakit daun tomat secara manual melalui pengamatan visual semata tanpa alat bantu identifikasi, sehingga bergantung sepenuhnya pada pengalaman pribadi atau konsultasi dengan penyuluh yang tidak selalu tersedia.

Selain itu, ditemukan bahwa daun tomat di lahan menunjukkan berbagai gejala fisik, seperti daun menggulung dengan permukaan yang terasa kasar, terdapat bercak coklat kehitaman pada permukaan daun, serta perubahan warna daun menjadi hijau kecoklatan. Tingkat keparahan penyakit tergolong sedang dengan penyebaran yang telah menjangkau beberapa tanaman. Faktor lingkungan berupa kelembapan tinggi dan curah hujan yang intens menjadi penyebab utama penyebaran penyakit, sementara tanaman yang berada pada fase vegetatif menunjukkan pertumbuhan yang terhambat. Kegiatan observasi ini menggunakan instrumen observasi yang terdapat pada lampiran 10.

Observasi tahap kedua dilakukan setelah aplikasi selesai dibangun, yaitu dengan mengujicobakan aplikasi secara langsung di lahan pertanian untuk mengamati kesesuaian hasil klasifikasi aplikasi dengan kondisi penyakit daun tomat yang ditemukan di lapangan. Hasil observasi tahap ini digunakan sebagai bahan evaluasi terhadap performa aplikasi dalam kondisi nyata.

3.2.2 Wawancara

Wawancara dilakukan secara langsung kepada petani tomat di Desa Babakanmulya sebagai narasumber utama. Berdasarkan hasil wawancara dengan Bapak Aep Saepudin selaku petani tomat, diperoleh informasi bahwa identifikasi penyakit daun tomat yang selama ini dilakukan masih mengandalkan pengamatan visual sederhana dan konsultasi informal. Petani mengungkapkan kesulitan nyata dalam membedakan jenis penyakit, dikarenakan gejala visual yang ditampilkan terlihat hampir serupa satu sama lain.

Upaya penanganan yang biasa dilakukan adalah dengan menyemprotkan pestisida yang tersedia di toko pertanian tanpa memastikan jenis penyakit secara akurat, sehingga penanganan tidak selalu efektif. Hal ini menyebabkan pertumbuhan tanaman terhambat, hasil panen berkurang, dan kualitas buah menurun. Petani juga menyatakan bahwa kehadiran aplikasi *Android* yang mampu mengidentifikasi penyakit daun tomat secara otomatis akan sangat membantu kegiatan pertanian sehari-hari. Wawancara dilaksanakan pada tanggal 30 Januari 2026 dan instrumen wawancara terdapat pada lampiran 10.

3.2.3 Kuisioner

Kuesioner digunakan untuk mengukur tingkat kepuasan pengguna terhadap aplikasi klasifikasi penyakit daun tomat yang telah dikembangkan. Kuesioner diberikan kepada responden setelah mereka mencoba menggunakan aplikasi secara langsung di lapangan. Penilaian dilakukan menggunakan skala dengan rentang 1 hingga 5, di mana nilai 1 berarti sangat tidak setuju dan nilai 5 berarti sangat setuju. Aspek yang dinilai meliputi kemudahan penggunaan , keakuratan hasil klasifikasi, tampilan antarmuka, kecepatan respon aplikasi, dan manfaat aplikasi secara keseluruhan bagi kegiatan pertanian petani. Instrumen kuesioner selengkapnya terdapat pada lampiran 11.

3.2.4 Studi Pustaka

Studi pustaka dilakukan dengan cara mengumpulkan data dari berbagai sumber seperti buku, jurnal ilmiah, artikel penelitian, dan sumber daring yang relevan dengan topik penelitian. Seiring meningkatnya penerapan teknologi kecerdasan buatan dalam bidang pertanian, algoritma *deep learning* khususnya *Convolutional Neural Network* (CNN) menjadi salah satu solusi yang banyak dikaji untuk mengatasi permasalahan identifikasi penyakit tanaman secara otomatis. CNN merupakan teknik pengolahan citra yang mampu

mengekstraksi fitur visual secara hierarkis sehingga dapat membedakan pola warna, tekstur, dan bentuk pada citra daun tanaman dengan tingkat akurasi yang tinggi.

Beberapa penelitian menunjukkan bahwa arsitektur *MobileNetV2* yang dikembangkan melalui pendekatan *transfer learning* efektif dalam meningkatkan akurasi klasifikasi penyakit tanaman, terutama pada *dataset* berukuran terbatas. Arsitektur ini juga terbukti unggul dalam efisiensi komputasi sehingga sesuai untuk diimplementasikan pada perangkat *mobile* seperti *smartphone Android*. Namun, penelitian yang secara spesifik mengembangkan aplikasi *Android* untuk klasifikasi penyakit daun tomat menggunakan data primer dari lapangan di wilayah Kabupaten Kuningan masih sangat terbatas. Oleh karena itu, diperlukan suatu sistem yang mampu mengklasifikasikan penyakit daun tomat secara otomatis sekaligus dapat diakses secara mandiri oleh petani melalui perangkat *Android*.

Studi pustaka dalam penelitian ini bersumber dari jurnal ilmiah nasional dan internasional yang berkaitan dengan klasifikasi citra penyakit tanaman menggunakan CNN, buku referensi mengenai budidaya tomat dan pengendalian hama terpadu yang diterbitkan oleh Kementerian Pertanian, serta dokumentasi teknis resmi dari *framework TensorFlow*, *TensorFlow Lite*, dan *platform* pengembangan *Android*. Selain itu, *dataset* citra daun tomat dari sumber daring seperti *PlantVillage*, *Kaggle*, dan *Google Dataset* juga digunakan sebagai data sekunder untuk melengkapi *dataset* primer yang dikumpulkan dari lapangan.

3.2.5 Jenis dan Sumber Data

Adapun jenis dan sumber data yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Data Primer

Data primer merupakan data yang diperoleh secara langsung dari sumber utama penelitian. Dalam penelitian ini, data primer diperoleh melalui kegiatan observasi dan wawancara dengan petani tomat di Desa Babakanmulya, Kecamatan Cigugur, Kabupaten Kuningan. Data yang diperoleh berupa informasi mengenai kondisi nyata penyakit daun tomat di lapangan, gejala-gejala yang tampak secara visual, kendala yang dihadapi petani dalam mengidentifikasi penyakit, serta kebutuhan terhadap sistem teknologi yang dapat membantu proses identifikasi. Data primer ini sangat penting karena menggambarkan kondisi terkini di lapangan yang menjadi dasar pengembangan sistem.

2. Data Sekunder

Data sekunder merupakan data yang diperoleh dari sumber tidak langsung, seperti buku, jurnal ilmiah, dan penelitian terdahulu yang relevan dengan topik penelitian. Data ini digunakan untuk mendukung dan memperkuat landasan teori, khususnya yang berkaitan dengan sistem pengolahan citra digital, algoritma *Convolutional Neural Network* (CNN), arsitektur *MobileNetV2*, serta *framework TensorFlow Lite* untuk implementasi model pada *platform Android*. Selain itu, data sekunder juga mencakup *dataset* citra daun tomat dari sumber daring yang berfungsi sebagai pelengkap data primer dalam pembentukan *dataset* pelatihan model klasifikasi.

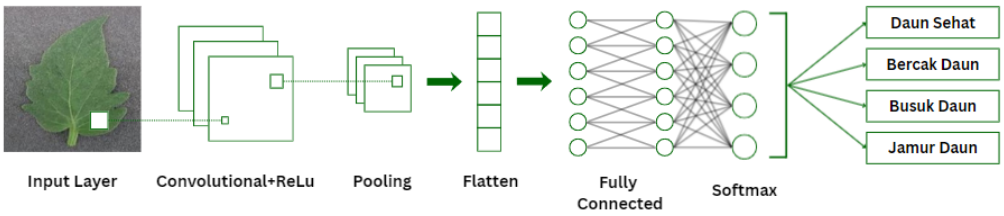
Berdasarkan sifatnya, data yang digunakan dalam penelitian ini adalah data kualitatif, yaitu data yang berupa deskripsi atau penjelasan yang menggambarkan kondisi sistem yang berjalan, permasalahan yang dihadapi petani, serta kebutuhan sistem yang akan dikembangkan. Data kualitatif ini diperoleh dari hasil observasi dan wawancara, kemudian dianalisis untuk menghasilkan solusi yang sesuai dengan kebutuhan pengguna, yaitu aplikasi *Android* berbasis CNN *MobileNetV2* yang mampu mengklasifikasikan penyakit pada daun tomat secara otomatis, akurat, dan mudah digunakan.

3.3 Metode Penyelesaian Masalah

Metode penyelesaian masalah merupakan teknik yang digunakan oleh peneliti untuk menyelesaikan masalah dalam proses penelitian. Dalam penelitian ini, metode penyelesaian masalah yang digunakan adalah algoritma *Convolutional Neural Network*.

3.3.1 Arsitektur Algoritma *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan arsitektur yang dapat dilatih dan terdiri dari beberapa lapisan. CNN menerima *input* berupa citra digital. Metode ini termasuk dalam penerapan *Deep Learning* yang berfungsi untuk mendeteksi dan mengenali objek pada sebuah gambar. Adapun struktur arsitektur CNN dapat dijelaskan sebagai berikut:



Gambar 3. 2 Arsitektur CNN

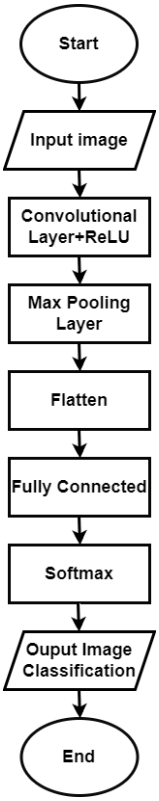
Penelitian ini difokuskan pada upaya untuk mengidentifikasi dan membedakan berbagai jenis penyakit pada daun tomat dengan memanfaatkan metode *Convolutional Neural Network* (CNN) sebagai pendekatan utama dalam proses klasifikasi.

Arsitektur CNN dimulai dari *input* berupa citra dengan resolusi 224x224 piksel. Citra ini kemudian diproses melalui tahap konvolusi menggunakan *filter* berukuran 3x3. Tahap konvolusi ini diimplementasikan melalui program Python dan menghasilkan hasil berupa *feature map. Feature map* memiliki resolusi yang lebih rendah dibandingkan citra awal, namun memiliki kedalaman yang lebih besar, karena telah mengekstraksi berbagai fitur penting dari citra *input*.

Setelah melalui proses konvolusi, citra hasil ekstraksi fitur akan dilanjutkan ke tahap berikutnya, yaitu *max pooling*. *Max pooling* merupakan

proses di mana nilai maksimum diambil dari setiap area kecil pada *feature map* , sesuai dengan ukuran *kernel* yang telah ditentukan. Tujuan dari tahap ini adalah untuk mereduksi dimensi data sekaligus mempertahankan fitur-fitur penting. Setelah proses *max pooling* selesai, hasilnya akan kembali diproses melalui lapisan konvolusi berikutnya. Langkah ini dilakukan secara berulang untuk memperdalam pemahaman model terhadap pola pada citra, sehingga dapat meningkatkan akurasi hasil prediksi.

Selanjutnya, proses memasuki tahap *flatten*, yaitu proses di mana citra hasil ekstraksi fitur diubah menjadi bentuk satu dimensi. Transformasi ini diperlukan agar data dapat diproses lebih lanjut oleh lapisan *fully connected* pada jaringan *Convolutional Neural Network* (CNN). Tahap *flatten* menjadi penghubung antara hasil ekstraksi fitur dengan proses klasifikasi akhir.



Gambar 3. 3 Flowchart Metode CNN

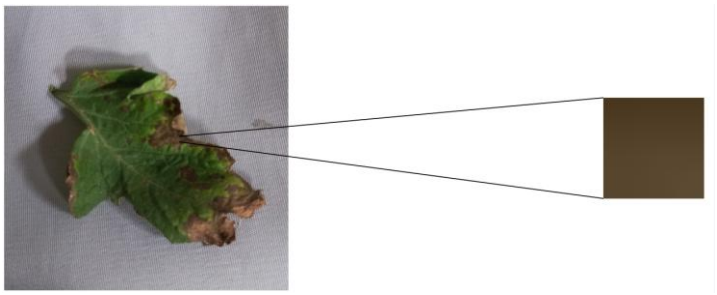
Berdasarkan gambar di atas, alur kerja CNN dimulai dari *Input Image*, kemudian diproses secara berurutan melalui *Convolutional Layer* + ReLU,

Max Pooling Layer, Flatten, Fully Connected, dan Softmax, hingga menghasilkan *Output Image Classification* identifikasi jenis penyakit pada daun tomat.

3.3.2 Perhitungan Tahapan Algoritma *Convolutional Neural Network*

Berikut merupakan proses tahapan perhitungan dari algoritma *Convolutional Neural Network* dibagi menjadi beberapa bagian yaitu*:*

1. ***Convolutional Layer***

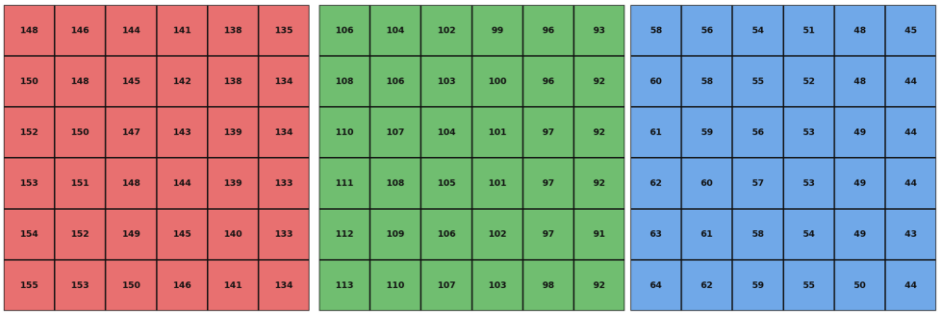


Gambar 3. 4 Sample Data *Input*an

Pada penelitian ini, proses komputasi algoritma *Convolutional Neural Network* (CNN) dimulai dari tahap *convolution*al *layer*. Lapisan ini merupakan komponen inti dari arsitektur CNN yang bertugas mengekstraksi fitur-fitur penting dari citra masukan. Proses ekstraksi dilakukan dengan cara menggeser sebuah *filter* atau *kernel* berukuran kecil di atas seluruh area citra, lalu mengalikan setiap elemen *kernel* dengan nilai piksel yang sejajar pada citra, kemudian menjumlahkan seluruh hasil perkalian tersebut untuk menghasilkan satu nilai keluaran. Nilai-nilai keluaran yang terkumpul dari seluruh posisi pergeseran *kernel* inilah yang membentuk apa yang disebut sebagai *feature map*. Kualitas dan kandungan informasi pada *feature map* sangat dipengaruhi oleh nilai-nilai bobot yang terdapat pada *kernel* yang digunakan, karena bobot tersebut menentukan pola seperti apa yang akan dikenali dan ditekankan dalam citra masukan.

Guna memperjelas alur perhitungan secara manual, digunakan contoh citra berukuran 6 × 6 piksel sebagai data sampel. Penggunaan

ukuran citra yang lebih kecil ini disengaja agar proses perhitungan dapat ditampilkan secara lengkap dan mudah dipahami, tanpa mengurangi esensi dari konsep yang mendasarinya. Data piksel yang digunakan merupakan data nyata yang diambil dari sampel *dataset* citra daun tomat kategori Busuk Daun pada penelitian ini. Setiap piksel dalam citra direpresentasikan ke dalam tiga saluran warna, yaitu saluran merah (Red), saluran hijau (Green), dan saluran biru (Blue), yang masing-masing memuat nilai intensitas piksel pada rentang 0 hingga 255. Ketiga matriks saluran warna tersebut telah ditampilkan pada sub-bab sebelumnya sebagai matriks 6 × 6 *RGB*.



Gambar 3. 5 Matriks 6x6 *Channel RGB*

Adapun *kernel* atau *filter* yang digunakan dalam proses konvolusi ini adalah operator Prewitt berukuran 3 × 3, sebagaimana ditunjukkan pada gambar berikut:

1	0	-1
1	0	-1
1	0	-1

Gambar 3. 6 *Kernel* 3x3

Operator Prewitt dipilih karena kemampuannya dalam mendeteksi gradien atau perubahan intensitas piksel yang merepresentasikan informasi tepi dan tekstur pada citra. *Kernel* ini bekerja dengan mengalikan bobot pada kolom kiri (+1), kolom tengah

(0), dan kolom kanan (−1), sehingga mampu menangkap perbedaan nilai piksel dari kiri ke kanan secara efektif (Arief Wisky & Sumijan, 2022). Proses konvolusi antara citra dan *filter* tersebut akan menghasilkan *feature map* dengan dimensi yang dapat dihitung menggunakan rumus berikut:

$$\boldsymbol{h = \frac{W - N + 2P}{s} + 1}$$

Dengan keterangan:

W = Ukuran gambar *input* (lebar atau tinggi)

N = Ukuran *kernel*

P = Jumlah *padding*

S = Nilai *stride* (langkah pergeseran *kernel*)

Berdasarkan rumus tersebut, dengan menggunakan citra berukuran 6 × 6, *kernel* 3 × 3, *stride* bernilai 1, dan tanpa penambahan *zero padding* (P = 0), maka dimensi *feature map* yang dihasilkan diperoleh melalui perhitungan h = (6 − 3 + 0) / 1 + 1 = 4. Artinya, setiap saluran warna akan menghasilkan *feature map* berukuran 4 × 4 yang memuat 16 nilai hasil konvolusi. Selanjutnya proses konvolusi pada masing-masing saluran warna dapat diilustrasikan sebagai berikut:

a. *Channel Red*

148	146	144	141	138	135
150	148	145	142	138	134
152	150	147	143	139	134
153	151	148	144	139	133
154	152	149	145	140	133
155	153	150	146	141	134

Gambar 3. 7 Matriks *Channel Red*

Proses konvolusi pada saluran merah (*Channel Red*) dilakukan dengan menggeser *kernel* 3 × 3 secara sistematis dari sudut kiri atas menuju sudut kanan bawah matriks piksel merah yang berukuran 6 × 6. Pada setiap posisi, elemen-elemen *kernel* dikalikan dengan nilai piksel yang berada di bawahnya, kemudian seluruh hasil perkalian tersebut dijumlahkan untuk menghasilkan satu nilai pada *feature map*. Proses ini diulang sebanyak 16 kali sesuai dengan ukuran *feature map* 4 × 4 yang dihasilkan. Adapun perhitungan manual untuk setiap posisi pergeseran dapat dilihat pada tabel berikut:

Tabel 3. 1 Perhitungan nilai *Red*

Proses 1	**Proses 2**
(148×1) + (146×0) + (144×-1) + (150×1) + (148×0) + (145×-1) + (152×1) + (150×0) + (147×-1) = 14	(146×1) + (144×0) + (141×-1) + (148×1) + (145×0) + (142×-1) + (150×1) + (147×0) + (143×-1) = 18
Proses 3	**Proses 4**
(144×1) + (141×0) + (138×-1) + (145×1) + (142×0) + (138×-1)	(141×1) + (138×0) + (135×-1) + (142×1) + (138×0) + (134×-1)

+ (147×1) + (143×0) + (139×-1) = 21	+ (143×1) + (139×0) + (134×-1) = 23
Proses 5	**Proses 6**
(150×1) + (148×0) + (145×-1) + (152×1) + (150×0) + (147×-1) + (153×1) + (151×0) + (148×-1) = 15	(148×1) + (145×0) + (142×-1) + (150×1) + (147×0) + (143×-1) + (151×1) + (148×0) + (144×-1) = 20
Proses 7	**Proses 8**
(145×1) + (142×0) + (138×-1) + (147×1) + (143×0) + (139×-1) + (148×1) + (144×0) + (139×-1) = 24	(142×1) + (138×0) + (134×-1) + (143×1) + (139×0) + (134×-1) + (144×1) + (139×0) + (133×-1) = 28
Proses 9	**Proses 10**
(152×1) + (150×0) + (147×-1) + (153×1) + (151×0) + (148×-1) + (154×1) + (152×0) + (149×-1) = 15	(150×1) + (147×0) + (143×-1) + (151×1) + (148×0) + (144×-1) + (152×1) + (149×0) + (145×-1) = 21
Proses 11	**Proses 12**
(147×1) + (143×0) + (139×-1) + (148×1) + (144×0) + (139×-1) + (149×1) + (145×0) + (140×-1) = 26	(143×1) + (139×0) + (134×-1) + (144×1) + (139×0) + (133×-1) + (145×1) + (140×0) + (133×-1) = 32
Proses 13	**Proses 14**
(153×1) + (151×0) + (148×-1) + (154×1) + (152×0) + (149×-1) + (155×1) + (153×0) + (150×-1) = 15	(151×1) + (148×0) + (144×-1) + (152×1) + (149×0) + (145×-1) + (153×1) + (150×0) + (146×-1) = 21
Proses 15	**Proses 16**
(148×1) + (144×0) + (139×-1) + (149×1) + (145×0) + (140×-1)	(144×1) + (139×0) + (133×-1) + (145×1) + (140×0) + (133×-1)

+ (150×1) + (146×0) + (141×-1) = 27	+ (146×1) + (141×0) + (134×-1) = 35

Feature map yang dihasilkan dari keseluruhan 16 proses konvolusi pada *Channel Red* di atas membentuk suatu array berukuran 4 × 4. Setiap nilai pada *feature map* ini merepresentasikan informasi gradien horizontal yang terdeteksi oleh operator Prewitt pada posisi piksel yang bersesuaian. Nilai positif menunjukkan peningkatan intensitas dari kanan ke kiri, sedangkan nilai negatif menunjukkan penurunan intensitas, dan nilai nol menandakan tidak adanya perubahan yang signifikan. Hasil *feature map Channel Red* dapat divisualisasikan sebagaimana ditunjukkan pada gambar berikut:

14	18	21	23
15	20	24	28
15	21	26	32
15	21	27	35

Gambar 3. 8 *Feature map Channel Red*

b. *Channel Green*

106	104	102	99	96	93
108	106	103	100	96	92
110	107	104	101	97	92
111	108	105	101	97	92
112	109	106	102	97	91
113	110	107	103	98	92

Gambar 3. 9 Matriks *Channel Green*

Tahap berikutnya adalah proses konvolusi pada saluran hijau (*Channel Green*). Sama seperti proses pada saluran merah, *kernel* Prewitt 3 × 3 digeser secara berurutan dari posisi kiri atas hingga kanan bawah pada matriks piksel hijau yang berukuran 6 × 6. Setiap posisi pergeseran menghasilkan satu nilai melalui operasi perkalian elemen-per-elemen antara *kernel* dengan blok piksel yang bersesuaian, dilanjutkan dengan penjumlahan seluruh hasil perkalian tersebut. Totalnya terdapat 16 proses konvolusi yang menghasilkan *feature map* berukuran 4 × 4. Rincian perhitungan manual untuk setiap proses konvolusi pada *Channel Green* dapat dilihat pada tabel berikut ini:

Tabel 3. 2 Perhitungan nilai *Green*

Proses 1	**Proses 2**
(106×1) + (104×0) + (102×-1) + (108×1) + (106×0) + (103×-1) + (110×1) + (107×0) + (104×-1) = 15	(104×1) + (102×0) + (99×-1) + (106×1) + (103×0) + (100×-1) + (107×1) + (104×0) + (101×-1) = 17
Proses 3	**Proses 4**
(102×1) + (99×0) + (96×-1)	(99×1) + (96×0) + (93×-1)

+ (103×1) + (100×0) + (96×-1) + (104×1) + (101×0) + (97×-1) = 20	+ (100×1) + (96×0) + (92×-1) + (101×1) + (97×0) + (92×-1) = 23
Proses 5	**Proses 6**
(108×1) + (106×0) + (103×-1) + (110×1) + (107×0) + (104×-1) + (111×1) + (108×0) + (105×-1) = 17	(106×1) + (103×0) + (100×-1) + (107×1) + (104×0) + (101×-1) + (108×1) + (105×0) + (101×-1) = 19
Proses 7	**Proses 8**
(103×1) + (100×0) + (96×-1) + (104×1) + (101×0) + (97×-1) + (105×1) + (101×0) + (97×-1) = 22	(100×1) + (96×0) + (92×-1) + (101×1) + (97×0) + (92×-1) + (101×1) + (97×0) + (92×-1) = 26
Proses 9	**Proses 10**
(110×1) + (107×0) + (104×-1) + (111×1) + (108×0) + (105×-1) + (112×1) + (109×0) + (106×-1) = 18	(107×1) + (104×0) + (101×-1) + (108×1) + (105×0) + (101×-1) + (109×1) + (106×0) + (102×-1) = 20
Proses 11	**Proses 12**
(104×1) + (101×0) + (97×-1) + (105×1) + (101×0) + (97×-1) + (106×1) + (102×0) + (97×-1) = 24	(101×1) + (97×0) + (92×-1) + (101×1) + (97×0) + (92×-1) + (102×1) + (97×0) + (91×-1) = 29
Proses 13	**Proses 14**
(111×1) + (108×0) + (105×-1) + (112×1) + (109×0) + (106×-1) + (113×1) + (110×0) + (107×-1) = 18	(108×1) + (105×0) + (101×-1) + (109×1) + (106×0) + (102×-1) + (110×1) + (107×0) + (103×-1) = 21
Proses 15	**Proses 16**
(105×1) + (101×0) + (97×-1)	(101×1) + (97×0) + (92×-1)

+ (106×1) + (102×0) + (97×-1) + (107×1) + (103×0) + (98×-1) = 26	+ (102×1) + (97×0) + (91×-1) + (103×1) + (98×0) + (92×-1) = 31

Hasil *feature map* yang diperoleh dari proses konvolusi pada *Channel Green* membentuk array berukuran 4 × 4 dengan pola distribusi nilai yang mencerminkan gradien intensitas warna hijau pada citra masukan. Nilai-nilai yang terdapat pada *feature map* ini merepresentasikan seberapa besar perubahan intensitas piksel hijau terjadi di setiap titik lokasi pada citra. Visualisasi *feature map Channel Green* ditunjukkan pada gambar berikut:

15	17	20	23
17	19	22	26
18	20	24	29
18	21	26	31

Gambar 3. 10 *Feature map Channel Green*

c. *Channel Blue*

58	56	54	51	48	45
60	58	55	52	48	44
61	59	56	53	49	44
62	60	57	53	49	44
63	61	58	54	49	43
64	62	59	55	50	44

Gambar 3. 11 Matriks *Channel Blue*

Proses konvolusi selanjutnya diterapkan pada saluran biru (*Channel Blue*). Mekanisme yang digunakan sepenuhnya sama dengan proses pada kedua saluran sebelumnya, yakni menggeser *kernel* Prewitt 3 × 3 dari sudut kiri atas menuju sudut kanan bawah matriks piksel biru 6 × 6. Sebanyak 16 operasi konvolusi dilakukan secara berurutan untuk menghasilkan *feature map* berukuran 4 × 4 yang mencerminkan karakteristik gradien intensitas warna biru. Tabel berikut menyajikan rincian perhitungan manual untuk setiap proses konvolusi pada *Channel Blue*:

Tabel 3. 3 Perhitungan nilai *Blue*

Proses 1	**Proses 2**
(58×1) + (56×0) + (54×-1) + (60×1) + (58×0) + (55×-1) + (61×1) + (59×0) + (56×-1) = 14	(56×1) + (54×0) + (51×-1) + (58×1) + (55×0) + (52×-1) + (59×1) + (56×0) + (53×-1) = 17
Proses 3	**Proses 4**
(54×1) + (51×0) + (48×-1) + (55×1) + (52×0) + (48×-1)	(51×1) + (48×0) + (45×-1) + (52×1) + (48×0) + (44×-1)

+ (56×1) + (53×0) + (49×-1) = 20	+ (53×1) + (49×0) + (44×-1) = 23
Proses 5	**Proses 6**
(60×1) + (58×0) + (55×-1) + (61×1) + (59×0) + (56×-1) + (62×1) + (60×0) + (57×-1) = 15	(58×1) + (55×0) + (52×-1) + (59×1) + (56×0) + (53×-1) + (60×1) + (57×0) + (53×-1) = 19
Proses 7	**Proses 8**
(55×1) + (52×0) + (48×-1) + (56×1) + (53×0) + (49×-1) + (57×1) + (53×0) + (49×-1) = 22	(52×1) + (48×0) + (44×-1) + (53×1) + (49×0) + (44×-1) + (53×1) + (49×0) + (44×-1) = 26
Proses 9	**Proses 10**
(61×1) + (59×0) + (56×-1) + (62×1) + (60×0) + (57×-1) + (63×1) + (61×0) + (58×-1) = 15	(59×1) + (56×0) + (53×-1) + (60×1) + (57×0) + (53×-1) + (61×1) + (58×0) + (54×-1) = 20
Proses 11	**Proses 12**
(56×1) + (53×0) + (49×-1) + (57×1) + (53×0) + (49×-1) + (58×1) + (54×0) + (49×-1) = 24	(53×1) + (49×0) + (44×-1) + (53×1) + (49×0) + (44×-1) + (54×1) + (49×0) + (43×-1) = 29
Proses 13	**Proses 14**
(62×1) + (60×0) + (57×-1) + (63×1) + (61×0) + (58×-1) + (64×1) + (62×0) + (59×-1) = 15	(60×1) + (57×0) + (53×-1) + (61×1) + (58×0) + (54×-1) + (62×1) + (59×0) + (55×-1) = 21
Proses 15	**Proses 16**
(57×1) + (53×0) + (49×-1) + (58×1) + (54×0) + (49×-1)	(53×1) + (49×0) + (44×-1) + (54×1) + (49×0) + (43×-1)

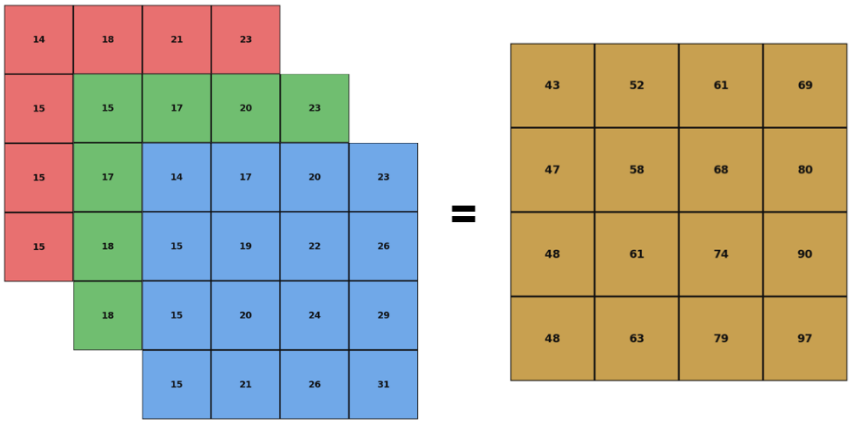
+ (59×1) + (55×0) + (50×-1) = 26	+ (55×1) + (50×0) + (44×-1) = 31

Berdasarkan hasil konvolusi pada *Channel Blue*, diperoleh *feature map* berukuran 4 × 4 yang menggambarkan distribusi gradien intensitas piksel biru pada citra masukan. Visualisasi *feature map Channel Blue* ditunjukkan pada gambar berikut:

14	17	20	23
15	19	22	26
15	20	24	29
15	21	26	31

Gambar 3. 12 *Feature map Channel Blue*

Feature map berukuran 4 × 4 yang dihasilkan secara terpisah dari ketiga saluran warna, yaitu *Channel Red*, *Channel Green*, dan *Channel Blue*, selanjutnya digabungkan dengan cara menjumlahkan nilai-nilai yang berada pada posisi yang sama. Proses penggabungan ini menghasilkan sebuah final *feature map* tunggal berukuran 4 × 4 yang merangkum informasi gradien dari ketiga saluran warna secara komprehensif. Hasil penjumlahan ketiga *feature map* tersebut dapat dilihat pada gambar berikut:



Gambar 3. 13 *Final Feature map* Hasil Konvolusi (R + G + B)

2. *ReLu (Rectified Linear Unit)*

Feature map yang dihasilkan dari proses konvolusi pada tahap sebelumnya selanjutnya diteruskan ke dalam lapisan fungsi aktivasi ReLU (*Rectified Linear Unit*). Fungsi aktivasi ini merupakan salah satu komponen krusial dalam arsitektur jaringan saraf tiruan yang berperan penting dalam memperkenalkan sifat non-linearitas ke dalam model. Tanpa kehadiran fungsi aktivasi non-linear seperti ReLU, seluruh rangkaian lapisan pada CNN secara matematis hanya akan setara dengan sebuah transformasi linear tunggal, sehingga kemampuan model dalam mempelajari pola-pola kompleks pada citra menjadi sangat terbatas.

Cara kerja ReLU pada dasarnya sangat sederhana namun sangat efektif. Fungsi ini bekerja dengan cara mengevaluasi setiap nilai pada *feature map* : apabila suatu nilai bersifat positif, maka nilai tersebut akan dipertahankan apa adanya tanpa mengalami perubahan apapun; sebaliknya, apabila suatu nilai bersifat negatif, maka nilai tersebut akan langsung diubah menjadi nol. Definisi matematis dari fungsi ReLU dinyatakan sebagai berikut:

$$\boldsymbol{f(x) = max(x, 0)}$$

Keberadaan nilai-nilai negatif pada *feature map* berpotensi menimbulkan permasalahan serius dalam proses pelatihan jaringan saraf tiruan. Secara khusus, nilai negatif dapat menyebabkan gradien pada sejumlah neuron mendekati nol, terutama ketika jaringan menerima *input* dengan nilai yang sangat besar. Kondisi ini mengarah pada permasalahan yang dikenal sebagai *vanishing gradient*, yakni suatu fenomena di mana sinyal gradasi yang digunakan untuk memperbarui bobot jaringan menjadi sangat kecil hingga praktis tidak memberikan kontribusi yang berarti terhadap proses pembelajaran. Akibatnya, kemampuan model dalam mengenali dan mempelajari pola-pola pada citra menjadi sangat terhambat. Dengan menerapkan ReLU dan mengubah seluruh nilai negatif menjadi nol, permasalahan ini dapat diatasi secara efisien sekaligus mempercepat konvergensi pelatihan model. Adapun penerapan fungsi aktivasi ReLU pada *final feature map* yang merupakan hasil penjumlahan ketiga saluran warna ditunjukkan pada gambar berikut:

43	52	61	69
47	58	68	80
48	61	74	90
48	63	79	97

Gambar 3. 14 Hasil Aktivasi ReLU

Berdasarkan gambar di atas, dapat diamati bahwa setelah proses aktivasi ReLU diterapkan pada final *feature map* berukuran 4 × 4, seluruh nilai yang sebelumnya bernilai negatif telah digantikan dengan nilai 0 (nol), sementara nilai-nilai yang sudah bernilai positif tetap dipertahankan sesuai besaran aslinya. Dari total 16 nilai pada final

feature map, sebagian besar bernilai nol karena tingginya keseragaman nilai piksel pada citra sampel, sedangkan nilai positif yang tersisa merepresentasikan keberadaan informasi gradien yang bermakna pada posisi-posisi tertentu. Hasil aktivasi ReLU ini kemudian menjadi masukan bagi tahap berikutnya dalam arsitektur CNN, yaitu *Pooling Layer*.

3. *Pooling Layer*

Pooling layer merupakan lapisan yang diposisikan setelah lapisan konvolusi beserta fungsi aktivasinya dalam struktur arsitektur CNN. Lapisan ini memiliki fungsi utama untuk melakukan reduksi dimensi spasial pada *feature map* yang dihasilkan oleh lapisan konvolusi, tanpa mengorbankan informasi-informasi penting yang telah berhasil diekstraksi sebelumnya. Dengan berkurangnya dimensi data, kompleksitas komputasi pada lapisan-lapisan berikutnya dapat ditekan secara signifikan, sehingga proses pelatihan model menjadi lebih efisien dari segi waktu maupun kebutuhan memori. Selain itu, *pooling layer* juga turut berkontribusi dalam meningkatkan ketahanan model terhadap variasi kecil pada posisi atau orientasi objek dalam citra, karena proses reduksi yang dilakukannya bersifat invarian terhadap pergeseran kecil (translation invariance).

Pada penelitian ini, jenis *pooling* yang diterapkan adalah *max pooling*. Teknik ini bekerja dengan cara membagi *feature map* menjadi blok-blok piksel berukuran 2 × 2, kemudian memilih dan mengambil nilai tertinggi dari setiap blok tersebut sebagai representasi keluaran. Pendekatan ini didasarkan pada asumsi bahwa nilai piksel tertinggi dalam suatu wilayah merepresentasikan fitur paling dominan dan paling relevan yang perlu dipertahankan. Proses pengambilan nilai maksimum ini dilakukan secara bertahap dengan menggeser blok 2 × 2 ke seluruh

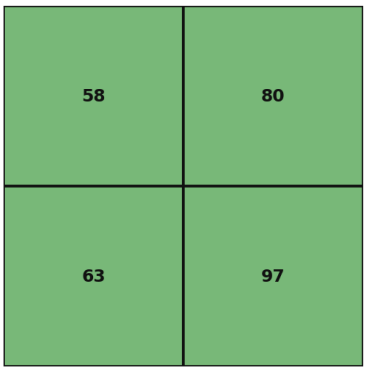
area *feature map* hingga seluruh wilayah telah diproses secara menyeluruh. Dari *feature map* berukuran 4 × 4 yang merupakan keluaran dari proses aktivasi ReLU, proses *max pooling* dengan blok 2 × 2 akan menghasilkan *feature map* baru berukuran 2 × 2. Perhitungan untuk setiap blok adalah sebagai berikut:

Blok 1 (baris 0–1, kolom 0–1): *max*(43, 52, 47, 58) = 58

Blok 2 (baris 0–1, kolom 2–3): *max*(61, 69, 68, 80) = 80

Blok 3 (baris 2–3, kolom 0–1): *max*(48, 61, 48, 63) = 63

Blok 4 (baris 2–3, kolom 2–3): *max*(74, 90, 79, 97) = 97



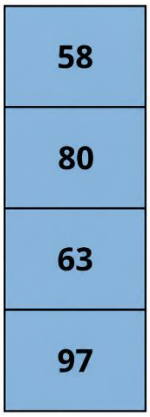
Gambar 3. 15 Hasil *Max Pooling* 2 × 2

Berdasarkan perhitungan *max pooling* di atas, diperoleh *feature map* baru berukuran 2 × 2 dengan nilai [58, 80; 63, 97]. Dimensi data telah berhasil direduksi dari 4 × 4 menjadi 2 × 2, yang berarti jumlah elemen berkurang dari 16 menjadi 4 nilai, namun informasi fitur yang paling dominan tetap terjaga. Hasil *max pooling* ini selanjutnya akan diproses melalui tahap *flattening* sebagai langkah berikutnya dalam arsitektur CNN.

4. *Flattening*

Setelah tahapan *max pooling* menghasilkan *feature map* yang telah direduksi dimensinya, langkah selanjutnya dalam arsitektur CNN adalah proses *flattening* atau perataan. Tahapan ini memiliki peranan yang sangat krusial karena berfungsi sebagai jembatan penghubung antara bagian ekstraksi fitur (*convolutional layer*, ReLU, dan *pooling*) dengan bagian klasifikasi (*Fully connected layer*). Tanpa tahap ini, data dalam bentuk matriks dua dimensi tidak dapat langsung diumpankan ke dalam jaringan saraf tiruan yang memerlukan *input* berupa vektor satu dimensi.

Secara operasional, *flattening* bekerja dengan cara merentangkan seluruh nilai yang tersimpan dalam *feature map* berdimensi n × m menjadi sebuah vektor tunggal berdimensi satu. Dengan mengacu pada hasil *max pooling* 2 × 2 yang menghasilkan matriks [58, 80; 63, 97], proses *flattening* akan mengonversinya menjadi sebuah vektor berukuran 4 × 1. Visualisasi proses tersebut ditunjukkan pada gambar berikut:



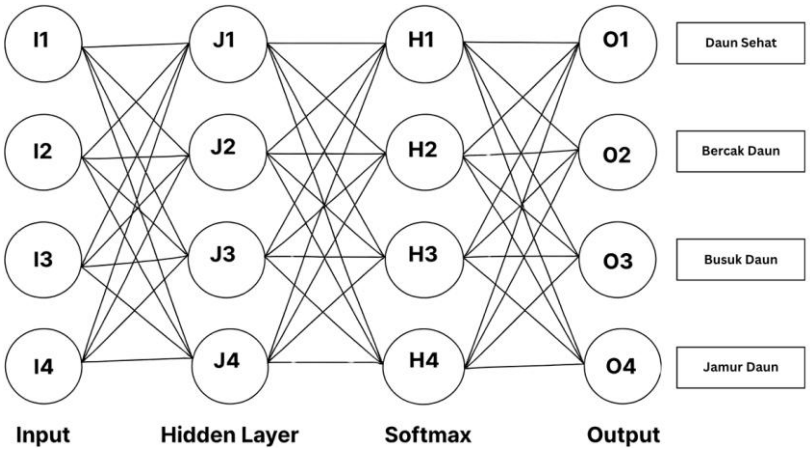
Gambar 3. 16 Proses *Flattening*

Sebagaimana yang ditunjukkan pada gambar di atas, keluaran dari proses *flattening* adalah sebuah vektor *input* berukuran 4 × 1 dengan nilai I1 = 58, I2 = 80, I3 = 63, dan I4 = 97. Keempat nilai ini merepresentasikan ringkasan fitur yang telah diekstraksi dari citra melalui serangkaian proses konvolusi, aktivasi ReLU, dan max *pooling*. Vektor satu dimensi ini kemudian akan digunakan sebagai masukan pada

lapisan *fully connected*, di mana setiap neuron pada vektor *input* akan terhubung secara penuh dengan seluruh neuron pada lapisan tersembunyi (*hidden layer*) untuk menentukan bobot-bobot yang akan digunakan dalam proses klasifikasi akhir.

5. *Fully connected layer*

Tahap terakhir dalam proses komputasi CNN adalah *Fully connected layer* atau lapisan yang sepenuhnya terhubung. Setelah melalui tahap *flattening* dan data berhasil dikonversi menjadi vektor satu dimensi, vektor tersebut digunakan sebagai masukan bagi lapisan ini. Karakteristik utama dari *Fully connected layer* adalah setiap jaringan pada lapisan ini terhubung secara langsung dan penuh dengan seluruh neuron yang ada pada lapisan sebelumnya, sehingga tidak ada satupun koneksi yang terputus.



Gambar 3. 17 Proses *Fully connected layer*

Fully connected layer digunakan untuk menentukan kelas fitur yang paling dominan dari citra yang dianalisis. *Input* dari *Fully connected layer* adalah vektor 4 × 1 hasil *flattening*, yang merupakan akumulasi hasil dari seluruh tahap pemrosesan sebelumnya. Berikut adalah perhitungan untuk setiap lapisan tersembunyi (*hidden layer*) menggunakan rumus:

$\sum$ Ii × vij = Ji (*Hidden Layer* Pertama)

$\sum$ Ji × wij = Hi (*Hidden Layer* Kedua)

$\sum$ Hi × xij = Oi (*Output layer*)

Dalam perhitungan ini, I1, I2, I3, dan I4 merupakan nilai-nilai *input* yang berasal dari hasil *flattening*, yaitu berturut-turut bernilai 58, 80, 63, dan 97. Nilai-nilai *input* tersebut kemudian dikalikan dengan bobot yang telah ditentukan untuk menghasilkan hidden *layer* pertama, yang direpresentasikan oleh neuron-neuron J1, J2, J3, dan J4. Adapun perhitungannya adalah sebagai berikut:

J1 = (58×0.2) + (80×0.2) + (63×0.2) + (97×0.2) = 59.6

J2 = (58×0.4) + (80×0.4) + (63×0.4) + (97×0.4) = 119.2

J3 = (58×0.3) + (80×0.3) + (63×0.3) + (97×0.3) = 89.4

J4 = (58×0.7) + (80×0.7) + (63×0.7) + (97×0.7) = 208.6

Hidden *layer* pertama yang terdiri atas neuron J1, J2, J3, dan J4 merepresentasikan transformasi pertama dari data *input* melalui proses perkalian dengan bobot yang berbeda-beda. Nilai-nilai J tersebut selanjutnya diteruskan dan dikalikan kembali dengan satu set bobot baru untuk membentuk hidden *layer* kedua yang direpresentasikan oleh neuron-neuron H1, H2, H3, dan H4, sebagai berikut:

H1 = (59.6×0.5) + (119.2×0.5) + (89.4×0.5) + (208.6×0.5) = 238.4

H2 = (59.6×0.8) + (119.2×0.8) + (89.4×0.8) + (208.6×0.8) = 381.44

H3 = (59.6×0.3) + (119.2×0.3) + (89.4×0.3) + (208.6×0.3) = 143.04

H4 = (59.6×0.6) + (119.2×0.6) + (89.4×0.6) + (208.6×0.6) = 286.08

Nilai-nilai yang dihasilkan dari hidden layer kedua, yaitu H1, H2, H3, dan H4, merepresentasikan abstraksi fitur pada tingkat yang lebih tinggi. Langkah selanjutnya adalah mengalikan setiap nilai tersebut

dengan satu set bobot terakhir untuk menghasilkan empat nilai pada lapisan keluaran (*Output layer*). Masing-masing nilai keluaran, yang dilambangkan dengan O1 hingga O4, merepresentasikan skor kecenderungan citra masukan terhadap salah satu dari empat kelas penyakit daun tomat yang menjadi target klasifikasi pada penelitian ini, yaitu Daun Sehat, Bercak Daun, Busuk Daun, dan Jamur Daun. Perhitungannya adalah sebagai berikut:

O1 = (238.4×0.011) + (381.44×0.011) + (143.04×0.011) + (286.08×0.011) = 1.1539

O2 = (238.4×0.08) + (381.44×0.08) + (143.04×0.08) + (286.08×0.08) = 0.8392

O3 = (238.4×0.019) + (381.44×0.019) + (143.04×0.019) + (286.08×0.019) = 1.993

O4 = (238.4×0.014) + (381.44×0.014) + (143.04×0.014) + (286.08×0.014) = 1.4685

Nilai-nilai keluaran O1 hingga O4 yang diperoleh di atas masih berupa skor mentah yang belum dapat diinterpretasikan secara langsung sebagai probabilitas. Oleh karena itu, langkah terakhir yang harus dilakukan adalah menerapkan fungsi aktivasi *Softmax* untuk mengkonversi skor-skor mentah tersebut menjadi distribusi probabilitas yang terstandarisasi. Fungsi *Softmax* bekerja dengan cara mengeksponensiasi setiap nilai keluaran, kemudian membaginya dengan jumlah total dari seluruh nilai yang telah dieksponensiasi. Rumus *Softmax* adalah sebagai berikut:

$$Softmax(O_i) = \frac{e^{O_i}}{\sum e^{O_j}}$$

Berikut hasil perhitungan probabilitas untuk setiap kelas penyakit:

S(O1) – Daun Sehat

$$= \frac{e^{1.1539}}{e^{1.1539} + e^{0.8392} + e^{1.993} + e^{1.4685}}$$

$$= \frac{3.1705}{3.1705 + 2.3145 + 7.3375 + 4.3427}$$

$$= \frac{3.1705}{17.1653} = 0.1847\ (18.47\%)$$

S(O2) – Bercak Daun

$$= \frac{e^{0.8392}}{e^{1.1539} + e^{0.8392} + e^{1.993} + e^{1.4685}}$$

$$= \frac{2.3145}{3.1705 + 2.3145 + 7.3375 + 4.3427}$$

$$= \frac{2.3145}{17.1653} = 0.1348\ (13.48\%)$$

S(O3) – Busuk Daun

$$= \frac{e^{1.993}}{e^{1.1539} + e^{0.8392} + e^{1.993} + e^{1.4685}}$$

$$= \frac{7.3375}{3.1705 + 2.3145 + 7.3375 + 4.3427}$$

$$= \frac{7.3375}{17.1653} = 0.4275\ (42.75\%)$$

S(O4) – Jamur Daun

$$= \frac{e^{1.4685}}{e^{1.1539} + e^{0.8392} + e^{1.993} + e^{1.4685}}$$

$$= \frac{4.3427}{3.1705 + 2.3145 + 7.3375 + 4.3427}$$

$$= \frac{4.3427}{17.1653} = 0.2530\ (25.30\%)$$

Berdasarkan hasil penerapan fungsi aktivasi *Softmax*, diperoleh nilai probabilitas untuk masing-masing kelas penyakit yang ditargetkan sebagai berikut:

S(O1) = 0.1847 (18.47%) → Daun Sehat

S(O2) = 0.1348 (13.48%) → Bercak Daun

S(O3) = 0.4275 (42.75%) → Busuk Daun

S(O4) = 0.2530 (25.30%) → Jamur Daun

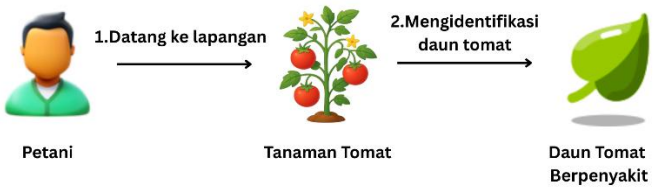
Dari seluruh nilai probabilitas yang diperoleh, kelas dengan probabilitas tertinggi adalah O3 yaitu Busuk Daun dengan nilai 0.4275 atau sebesar 42.75%. Hal ini berarti bahwa model CNN memprediksi citra daun tomat yang diujikan termasuk ke dalam kategori Busuk Daun. Dengan demikian, seluruh rangkaian proses komputasi CNN mulai dari *convolutional layer*, ReLU, *pooling layer*, *flattening*, hingga *Fully connected layer* dengan fungsi aktivasi *Softmax* telah berhasil diselesaikan dan menghasilkan keputusan klasifikasi yang terukur.

3.4 Perancangan

Perancangan sistem dalam penelitian ini mencakup analisis terhadap kondisi proses yang berjalan saat ini di Desa Babakanmulya serta rancangan sistem usulan yang dikembangkan sebagai solusi atas permasalahan yang ditemukan. Kedua analisis tersebut diuraikan pada rincian berikut.

3.4.1 Analisa Sistem Berjalan

Analisis sistem yang sedang berjalan adalah penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan dan hambatan-hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan. Adapun sistem yang sedang berjalan akan dijelaskan pada gambar berikut:



Gambar 3. 18 *Rich Picture* Sistem berjalan

Pada gambar di atas, penjelasan dari analisa sistem yang sedang berjalan yaitu sebagai berikut:

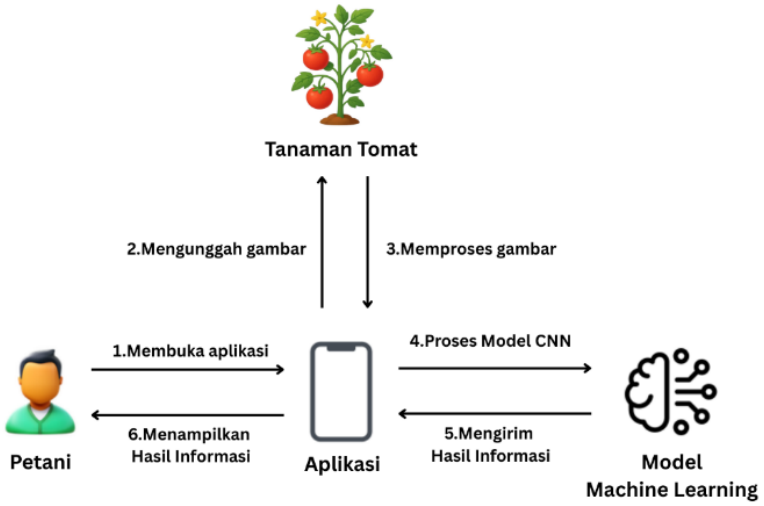
1. Petani datang ke lahan pertaniannya kemudian mengamati kondisi fisik daun tomat secara langsung.
2. Apabila ditemukan gejala penyakit pada daun, petani mencatat gejala tersebut berdasarkan pengamatan visual semata.
3. Petani memantau kondisi tanaman setelah penanganan dan mencatat hasilnya secara manual.

Saat ini, proses yang digunakan masih dilakukan secara manual hal ini berisiko menimbulkan kesalahan informasi, terutama apabila petani tidak sepenuhnya memahami seluruh jenis penyakit yang ada. Di sisi lain, tidak adanya sistem digital menyebabkan data kondisi tanaman tidak terdokumentasi dengan baik, sehingga menyulitkan dalam pengelolaan serta membatasi pengembangan layanan menuju arah yang lebih modern. Jika metode ini terus dipertahankan, maka peningkatan kualitas penanganan penyakit dan kemampuan beradaptasi dengan tuntutan era digital akan terhambat.

3.4.2 Analisa Sistem Usulan

Guna mengatasi permasalahan dalam proses klasifikasi penyakit daun tomat di Desa Babakanmulya, Kecamatan Cigugur, Kabupaten Kuningan, dikembangkan sebuah aplikasi berbasis Android yang dapat membantu petani maupun masyarakat umum dalam mengenali berbagai jenis penyakit pada daun

tomat dengan lebih mudah dan tepat. Rancangan sistem yang akan dibangun dapat dilihat pada Gambar 3.19 berikut:



Gambar 3. 19 *Rich Picture* Sistem Usulan

Pada gambar di atas, sistem aplikasi klasifikasi penyakit pada daun tomat yang dibangun yaitu sebagai berikut:

1. Petani membuka aplikasi klasifikasi penyakit daun tomat pada perangkat Android miliknya.
2. Petani dapat memilih mengambil gambar secara langsung menggunakan kamera atau mengambil gambar dari galeri, selanjutnya gambar akan diunggah ke aplikasi.
3. Aplikasi menerima dan memproses gambar daun tomat yang telah di*input*.
4. Aplikasi kemudian menganalisis objek menggunakan algoritma CNN MobileNetV2 yang tersimpan dalam format TensorFlow Lite dan berjalan secara lokal di dalam perangkat Android.
5. Setelah analisis selesai, aplikasi memberikan informasi mengenai jenis penyakit daun tomat yang terdeteksi beserta nilai probabilitasnya.
6. Aplikasi selanjutnya akan menampilkan hasil klasifikasi penyakit daun tomat beserta deskripsi singkat penyakit kepada petani.

Sistem yang diusulkan pada gambar diatas merupakan solusi berbasis aplikasi Android yang dirancang untuk mengatasi permasalahan dalam proses identifikasi dan klasifikasi penyakit pada daun tomat. Aplikasi ini memanfaatkan algoritma *Convolutional Neural Network* (CNN), untuk mengklasifikasikan gambar daun tomat secara otomatis. Dengan sistem ini, petani tidak lagi bergantung sepenuhnya pada penyuluh atau pengalaman pribadi untuk mendapatkan informasi mengenai jenis penyakit yang menyerang tanaman tomatnya. Petani cukup membuka aplikasi, mengambil atau mengunggah gambar daun tomat yang menunjukkan gejala, lalu sistem akan memproses gambar tersebut dan menampilkan hasil klasifikasi secara langsung. Informasi yang ditampilkan berupa nama jenis penyakit daun tomat yang terdeteksi beserta nilai probabilitas dan deskripsi singkat mengenai penyakit tersebut. Solusi ini tidak hanya mempermudah petani dalam proses identifikasi, tetapi juga meningkatkan efisiensi, serta memungkinkan akses informasi yang lebih luas dan mandiri bagi seluruh petani maupun masyarakat umum.

3.4.3 Fungsional dan Non-Fungsional

Berikut merupakan penjelasan dari kebutuhan fungsional dan non fungsional:

3.4.3.1 Fungsional

Analisis kebutuhan fungsional dilakukan untuk menentukan dan mendefinisikan fitur-fitur utama yang wajib tersedia dalam aplikasi klasifikasi penyakit daun tomat. Kebutuhan ini menjadi dasar dalam proses perancangan dan pengembangan sistem agar aplikasi yang dibangun dapat memenuhi kebutuhan pengguna secara optimal. Adapun kebutuhan fungsional pada aplikasi ini adalah sebagai berikut:

1. Aplikasi mampu menangkap gambar daun tomat secara langsung melalui kamera perangkat mobile maupun memilih gambar dari galeri.

2. Aplikasi dapat memproses citra yang diberikan dan melakukan klasifikasi jenis penyakit daun tomat menggunakan algoritma *Convolutional Neural Network* (CNN).
3. Aplikasi dapat menampilkan hasil klasifikasi berupa nama jenis penyakit daun tomat beserta informasi tambahan yang relevan terkait penyakit tersebut.

3.4.3.2 Non-Fungsional

Analisis kebutuhan non-fungsional merupakan gambaran mengenai kebutuhan pendukung di luar fungsi utama sistem yang diperlukan agar aplikasi dapat berjalan dengan baik. Kebutuhan ini mencakup aspek perangkat keras, perangkat lunak, serta karakteristik pengguna yang akan menggunakan aplikasi. Adapun kebutuhan non-fungsional pada aplikasi ini adalah sebagai berikut:

1. Kebutuhan Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam proses pengembangan dan pengujian aplikasi adalah sebagai berikut:

a. Unit Komputer/Laptop:
   - Prosesor: AMD Ryzen 4650U
   - RAM: 16 GB
   - Penyimpanan: SSD 256 GB
b. Smartphone untuk Pengujian:
   - RAM: minimal 4 GB
   - Penyimpanan: minimal 32 GB
   - Sistem Operasi: Android 10 atau yang setara

2. Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk mendukung proses pengembangan aplikasi meliputi:

- Sistem Operasi: Windows 11 (64-bit)
- Tools pengembangan: Google Colab, Visual Studio Code, Python, TensorFlow, Flutter

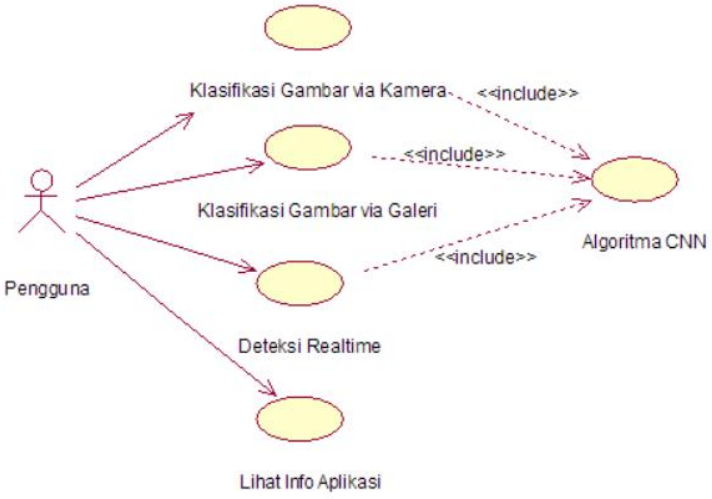
- Tools perancangan sistem: Draw.io
- Tools desain antarmuka: Figma

3.4.4 Perancangan Sistem Usulan

Perancangan sistem usulan bertujuan untuk mendeskripsikan secara rinci bagaimana aplikasi klasifikasi penyakit pada daun tomat yang akan dibuat dengan menggunakan UML.

3.4.4.1 *Use Case* Diagram

Pada aplikasi ini terdapat satu aktor utama yaitu pengguna yang berinteraksi langsung dengan sistem. Sistem tidak memerlukan manajemen multi-pengguna maupun hak akses bertingkat karena aplikasi bersifat personal dan dioperasikan langsung di perangkat milik petani. Berdasarkan hasil analisis kebutuhan, sistem ini memiliki lima *use case* utama yaitu: Ambil Gambar, Pilih Gambar, Klasifikasi Gambar, Deteksi *Realtime*, dan Lihat Info Aplikasi. Keempat *Use case* tersebut memiliki relasi dengan Algoritma CNN, yang berarti setiap *use case* secara wajib bergantung pada algoritma CNN dalam penggunaannya. Sedangkan *use case* Lihat Info Aplikasi memiliki tidak ketergantungan terhadap algoritma CNN dalam konteks menampilkan informasi aplikasi.



Gambar 3. 20 *Use Case Diagram* Aplikasi

3.4.4.2 Skenario Diagram

Berikut adalah uraian skenario untuk masing-masing *Use Case* pada sistem aplikasi:

a. Skenario Klasifikasi Gambar via Kamera

Tabel 3. 4 Skenario *Use Case* Klasifikasi Gambar via Kamera

Identifikasi	
ID Use Case	UC-01
Nama Use Case	Deteksi Gambar via Kamera
Tujuan	Memungkinkan pengguna untuk mendapatkan citra daun tomat baik melalui kamera secara langsung.
Deskripsi	
Prioritas	Primary, Essential
Aktor	Pengguna
Skenario Utama	
Pre-Condition	Pengguna telah masuk di menu utama
Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol Ambil via Kamera pada menu utama.	2. Sistem membuka antarmuka kamera.
3. Pengguna mengarahkan kamera dan mengambil gambar.	4. Sistem mengambil gambar dan menampilkannya pada layar pratinjau konfirmasi.
5. Pengguna menekan tombol centang untuk konfirmasi	6. Sistem melakukan pemrosesan gambar.
	7. Sistem menampilkan halaman hasil klasifikasi

Post Condition: Gambar daun tomat berhasil diperoleh dan siap diproses. Jika gambar objek tidak jelas maka akan menampilkan teks "**Gambar Objek Tidak Terdeteksi**".

b. Skenario Klasifikasi Gambar via Galeri

Tabel 3. 5 Skenario *Use Case* Klasifikasi Gambar via Galeri

Identifikasi	
ID Use Case	UC-02
Nama Use Case	Deteksi Gambar via Galeri
Tujuan	Memungkinkan pengguna untuk mendapatkan citra daun tomat baik melalui pemilihan gambar dari galeri.
Deskripsi	
Prioritas	Primary, Essential
Aktor	Pengguna
Skenario Utama	
Pre-Condition	Pengguna telah masuk di menu utama
Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol *Upload* dari Galeri pada menu utama.	2. Sistem membuka antarmuka *file picker* galeri.
3. Pengguna memilih salah satu foto dari galeri perangkat.	4. Sistem memuat gambar yang dipilih dan melakukan pemrosesan gambar.
	5. Sistem menampilkan halaman hasil klasifikasi
Post Condition: Gambar daun tomat berhasil diperoleh dan siap diproses. Jika gambar objek tidak jelas maka akan menampilkan teks "**Gambar Objek Tidak Terdeteksi**".	

c. Skenario Deteksi *Realtime*

Tabel 3. 6 Skenario Use Case Deteksi *Realtime*

Identifikasi	
ID Use Case	UC-04
Nama Use Case	Deteksi *Realtime*
Tujuan	Melakukan proses klasifikasi kondisi daun tomat secara kontinu dan langsung melalui pratinjau kamera
Deskripsi	
Prioritas	Primary, Essential
Aktor	Pengguna
Skenario Utama	
Pre-Condition	Pengguna telah masuk di menu utama
Aktor	**Reaksi Sistem**
1. Pengguna menekan menu 'Deteksi *Realtime*' pada halaman utama.	2. Sistem mengaktifkan kamera perangkat dan menampilkan pratinjau kamera secara langsung.
3. Pengguna mengarahkan kamera ke daun tomat yang ingin dideteksi.	4. Sistem menampilkan hasil deteksi untuk daun yang diarahkan.
Post Condition: Deteksi *realtime* berjalan dengan menampilkan hasil klasifikasi secara langsung. Jika objek tidak jelas maka akan menampilkan teks "**Objek Tidak Terdeteksi**".	

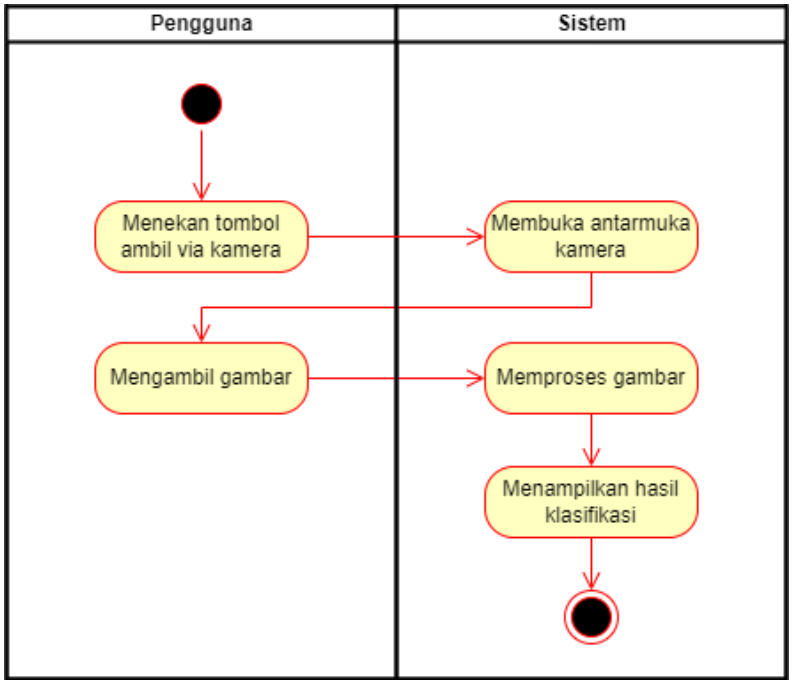
d. Skenario Lihat Info Aplikasi

Tabel 3. 7 Skenario *Use Case* Lihat Info Aplikasi

Identifikasi	
ID Use Case	UC-05
Nama Use Case	Lihat Info Aplikasi
Tujuan	Memberikan akses kepada pengguna untuk melihat informasi mengenai cara penggunaan aplikasi serta informasi tentang pengembang.
Deskripsi	
Prioritas	Primary, Essential
Aktor	Pengguna
Skenario Utama	
Pre-Condition	Pengguna telah masuk di menu utama
Aktor	**Reaksi Sistem**
1. Pengguna menekan menu *about* atau *history* pada halaman utama.	2. Sistem menampilkan halaman informasi riwayat dan informasi pengembang.
Post Condition: Pengguna berhasil mengakses panduan dan informasi aplikasi.	

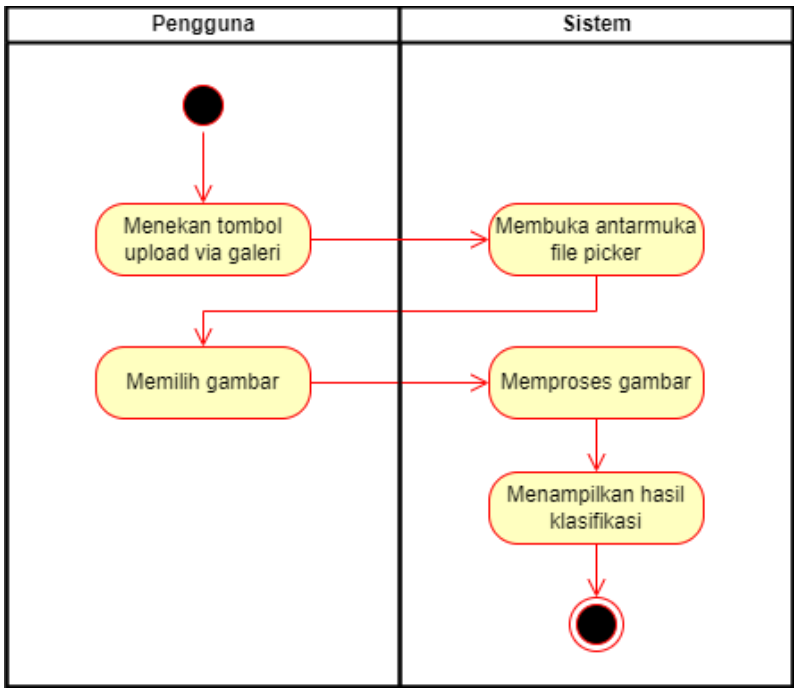
3.4.4.3 *Activity* Diagram

a. *Activity* Klasifikasi Gambar via Kamera



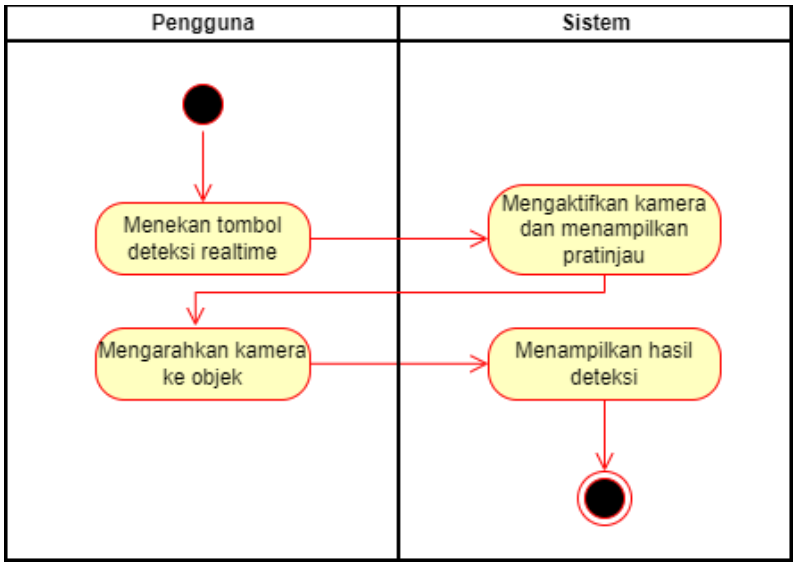
Gambar 3. 21 *Activity* Klasifikasi Gambar via Kamera

b. *Activity* Klasifikasi Gambar via Galeri



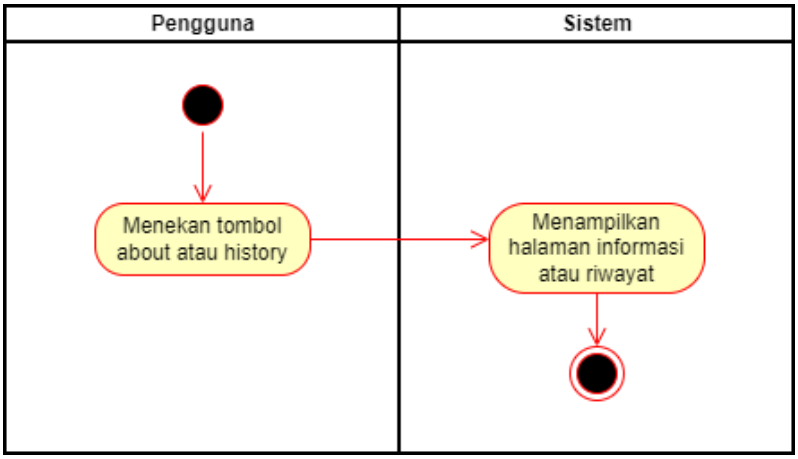
Gambar 3. 22 *Activity* Klasifikasi Gambar via Galeri

c. *Activity* Deteksi *Realtime*



Gambar 3. 23 *Activity* Deteksi *Realtime*

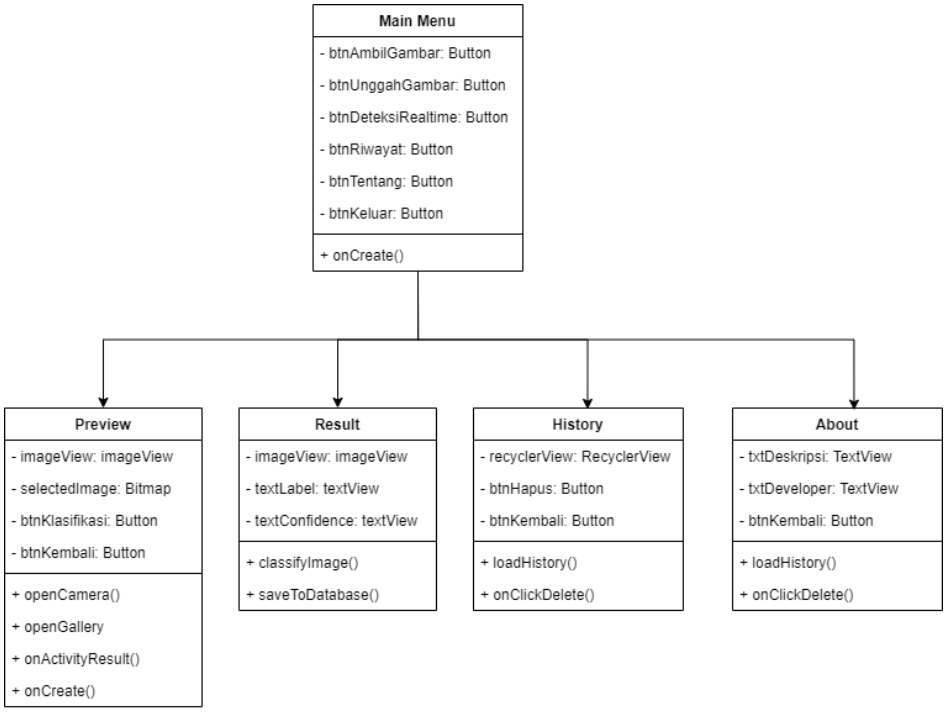
d. *Activity* Lihat Info Aplikasi



Gambar 3. 24 *Activity* Lihat Info Aplikasi

3.4.4.4 *Class Diagram*

Class Diagram merupakan diagram struktural dalam UML yang menggambarkan kelas-kelas yang terdapat pada sistem. Berikut adalah gambar *class diagram* aplikasi:



Gambar 3. 25 *Class Diagram*

Berikut merupakan tabel penjelasan dari *class* diagram pada gambar diatas:

Tabel 3. 8 Penjelasan *Class Diagram*

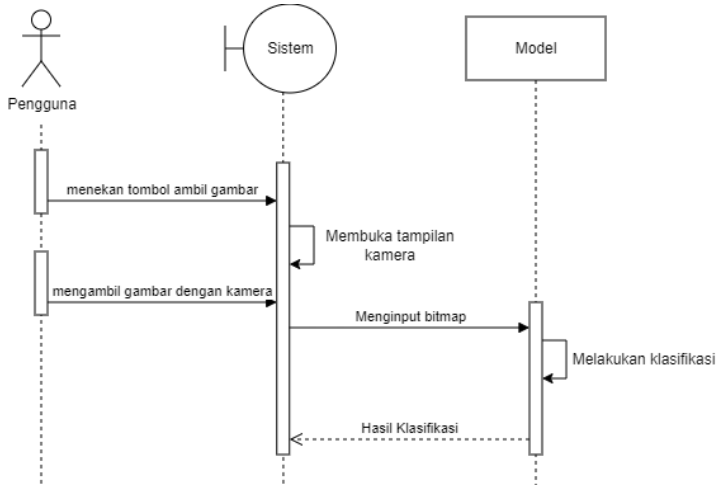
No	Nama Kelas	Atribut	Deskirpsi
1.	*MainMenu*	*- btnAmbilGambar: Button* *-btnUnggahGambar: Button* *- btnDeteksiRealtime: Button* *- btnRiwayat: Button* *- btnTentang: Button* *- btnKeluar: Button* *+ onCreate()*	Kelas utama (halaman beranda) tempat pengguna memilih fitur utama aplikasi seperti klasifikasi gambar, riwayat, deteksi *realtime*, dan tentang aplikasi.

2.	*Preview*	- *imageView: ImageView* - *selectedImage: Bitmap* - *btnKlasifikasi: Button* - *btnKembali: Button* + *openCamera()* + *openGallery()* + *onActivityResult()* + *onCreate()*	Kelas ini menangani tampilan gambar dari kamera atau galeri. Setelah pengguna memilih gambar, bisa melanjutkan untuk diklasifikasi.
3.	*Result*	- *imageView: ImageView* - *textLabel: TextView* - *textConfidence: TextView* + *classifyImage()* + *saveToDatabase()*	Kelas ini menampilkan hasil klasifikasi gambar berupa *label* jenis penyakit dan tingkat kepercayaannya. Juga menyimpan hasil klasifikasi ke *database* lokal.
4.	*History*	- *recyclerView: RecyclerView* - *btnHapus: Button* - *btnKembali: Button* + *loadHistory()* + *onClickDelete()*	Menampilkan daftar riwayat klasifikasi yang telah dilakukan sebelumnya. Pengguna dapat menghapus

			riwayat klasifikasi.
5.	*About*	- *txtDeskripsi: TextView* - *txtDeveloper: TextView* - *btnKembali: Button* + *getData()*	Menampilkan informasi tentang aplikasi, deskripsi singkat, dan nama pengembang.

3.4.4.5 *Sequence* Diagram

a. *Sequence* Klasifikasi Gambar via Kamera



Gambar 3. 26 *Sequence* Diagram Klasifikasi Gambar via Kamera

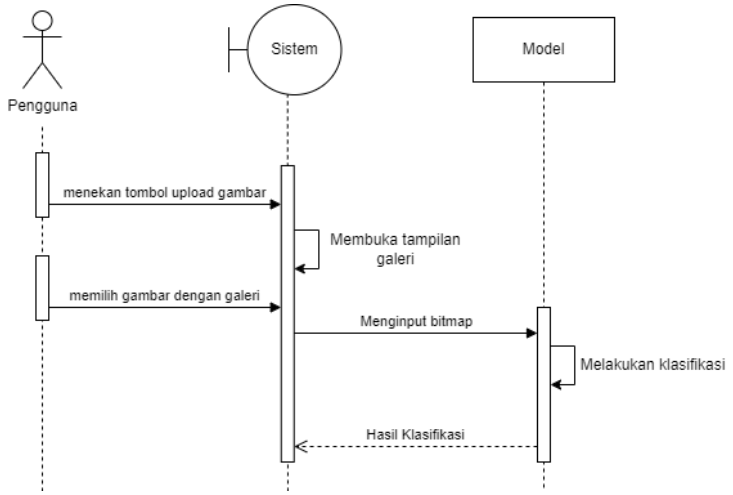
Berikut merupakan tabel penjelasan mengenai *sequence* diagram ambil gambar:

Tabel 3. 9 *Sequence* Klasifikasi Gambar via Kamera

No	Aktor / Objek	Pesan / Aksi	Deskripsi

1.	Pengguna	Menekan tombol ambil gambar	Pengguna memulai proses pengambilan gambar dengan menekan tombol Ambil via Kamera pada menu utama.
2.	Sistem	Membuka tampilan kamera	Sistem membuka antarmuka kamera perangkat sehingga pengguna dapat mengambil gambar secara langsung.
3.	Kamera	Menampilkan pratinjau kamera	Kamera perangkat menampilkan pratinjau gambar secara langsung kepada pengguna sebagai panduan pengambilan gambar.
4.	Pengguna	Mengambil gambar dengan kamera	Pengguna mengarahkan kamera ke daun tomat dan menekan tombol rana untuk mengambil gambar.
5.	Sistem	Meng*input* bitmap	Gambar hasil tangkapan kamera dikonversi menjadi format bitmap dan dimasukkan ke dalam sistem untuk diproses lebih lanjut.
6.	Kamera (Sistem)	Melakukan klasifikasi	Sistem mengirimkan bitmap ke model CNN untuk diproses dan diklasifikasikan sesuai kondisi daun tomat.
7.	Sistem	Hasil Klasifikasi	Sistem menampilkan hasil klasifikasi berupa nama kondisi daun tomat beserta nilai kepercayaan kepada pengguna.

b. *Sequence* Klasifikasi Gambar via Galeri



Gambar 3. 27 *Sequence* Diagram Klasifikasi Gambar via Galeri

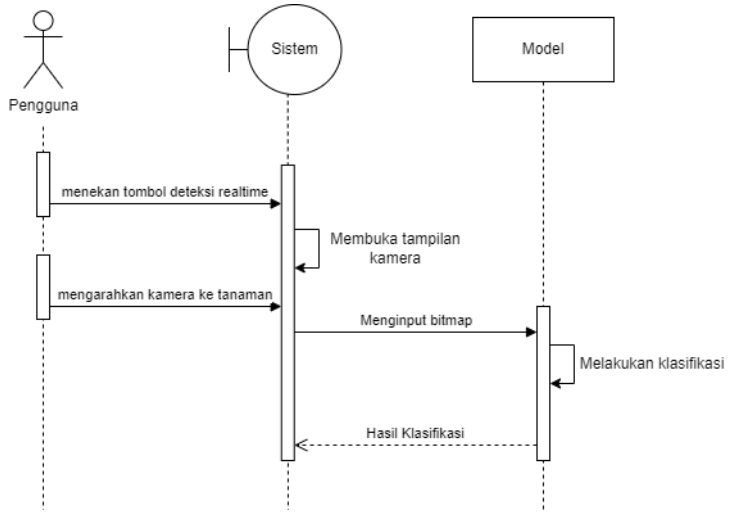
Berikut merupakan tabel penjelasan mengenai *sequence* diagram pilih gambar:

Tabel 3. 10 *Sequence* Klasifikasi Gambar via Galeri

No	Aktor / Objek	Pesan / Aksi	Deskripsi
1.	Pengguna	Menekan tombol unggah gambar	Pengguna memulai proses klasifikasi dengan memilih tombol *Upload* dari Galeri pada menu utama aplikasi.
2.	Sistem	Membuka tampilan galeri	Sistem membuka antarmuka file picker galeri perangkat untuk memungkinkan pengguna memilih gambar.
3.	Galeri	Menampilkan galeri perangkat	Galeri perangkat menampilkan daftar foto yang tersedia sehingga pengguna dapat memilih gambar daun tomat yang akan dianalisis.
4.	Pengguna	Memilih gambar dari galeri	Pengguna memilih salah satu foto daun tomat dari galeri perangkat untuk diproses oleh sistem.

5.	Sistem	Meng*input* bitmap	Gambar yang dipilih oleh pengguna dikonversi menjadi format bitmap dan dimasukkan ke dalam sistem untuk diproses lebih lanjut.
6.	Galeri (Sistem)	Melakukan klasifikasi	Bitmap dikirimkan ke model CNN untuk proses klasifikasi kondisi daun tomat.
7.	Sistem	Hasil Klasifikasi	Sistem menampilkan hasil klasifikasi berupa nama kondisi dan nilai kepercayaan kepada pengguna.

c. *Sequence* Deteksi *Realtime*



Gambar 3. 28 *Sequence* Diagram Deteksi *Realtime*

Berikut merupakan tabel penjelasan mengenai *sequence* diagram klasifikasi gambar:

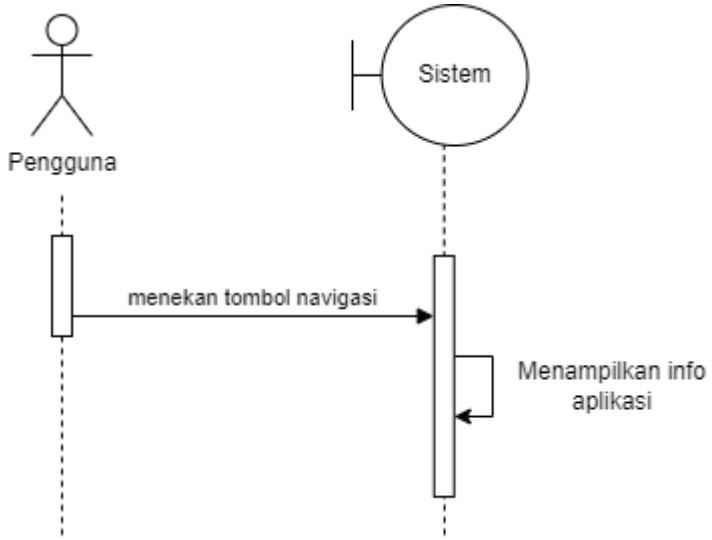
Tabel 3. 11 *Sequence* Klasifikasi Gambar

No	Aktor / Objek	Pesan / Aksi	Deskripsi

1.	Pengguna	Menekan tombol deteksi *realtime*	Pengguna memulai proses dengan menekan menu Deteksi *Realtime* pada halaman utama aplikasi.
2.	Sistem	Membuka tampilan kamera	Sistem merespons dengan mengaktifkan kamera perangkat dan menampilkan antarmuka kamera secara langsung ke layar.
3.	Pengguna	Mengarahkan perangkat ke tanaman	Pengguna mengarahkan kamera perangkat ke objek daun tomat yang ingin dideteksi kondisinya.
4.	Sistem	Meng*input* bitmap	Kamera mengambil citra *frame* secara kontinu dan mengirimkannya dalam bentuk bitmap ke sistem untuk diproses.
5.	Model CNN	Melakukan klasifikasi	Sistem mengirim bitmap ke model klasifikasi CNN untuk memproses dan mengenali kondisi daun tomat secara kontinu.
6.	Sistem	Menampilkan Hasil Klasifikasi	Model mengembalikan hasil klasifikasi berupa *label* kondisi dan nilai kepercayaan yang kemudian diteruskan ke antarmuka pengguna.
7.	Sistem	Hasil Klasifikasi	Sistem menampilkan hasil deteksi secara langsung pada layar pratinjau kamera. Jika objek tidak terdeteksi, sistem

			menampilkan teks "Objek Tidak Terdeteksi".

d. *Sequence* Lihat Info Aplikasi



Gambar 3. 29 *Sequence* Diagram Lihat Info Aplikasi

Berikut merupakan tabel penjelasan mengenai *sequence* diagram lihat info aplikasi:

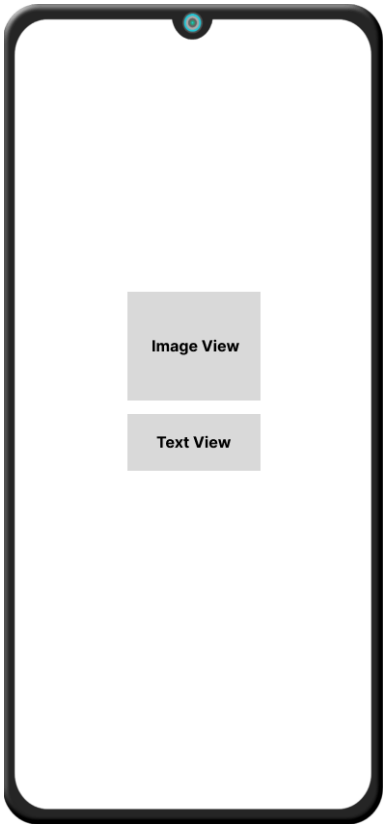
Tabel 3. 12 *Sequence* Lihat Info Aplikasi

No	Aktor / Objek	Pesan / Aksi	Deskripsi
1.	Pengguna	Menekan tombol tentang aplikasi	Pengguna memilih menu About atau Tentang pada halaman utama untuk mengakses informasi mengenai aplikasi.
2.	Sistem	Menampilkan informasi tentang aplikasi	Sistem menampilkan halaman informasi yang memuat deskripsi singkat aplikasi, panduan penggunaan, dan informasi pengembang.

3.	Pengguna	Menekan tombol riwayat	Pengguna memilih menu Riwayat untuk melihat daftar hasil klasifikasi yang pernah dilakukan sebelumnya.
4.	Sistem	Menampilkan daftar riwayat klasifikasi	Sistem menampilkan daftar hasil klasifikasi yang telah disimpan di *database* lokal kepada pengguna.

3.4.5 Perancangan Tampilan

3.4.5.1 Rancangan Antarmuka *Splash screen*



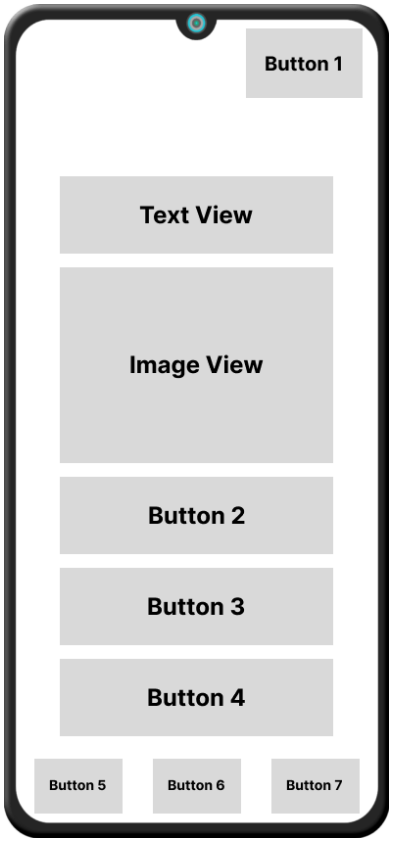
Gambar 3. 30 *Splash screen*

Berdasarkan gambar 3.30 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Image View	Menampilkan logo aplikasi

2	Text View	Menampilkan nama aplikasi

3.4.5.2 Rancangan Antarmuka Menu Utama



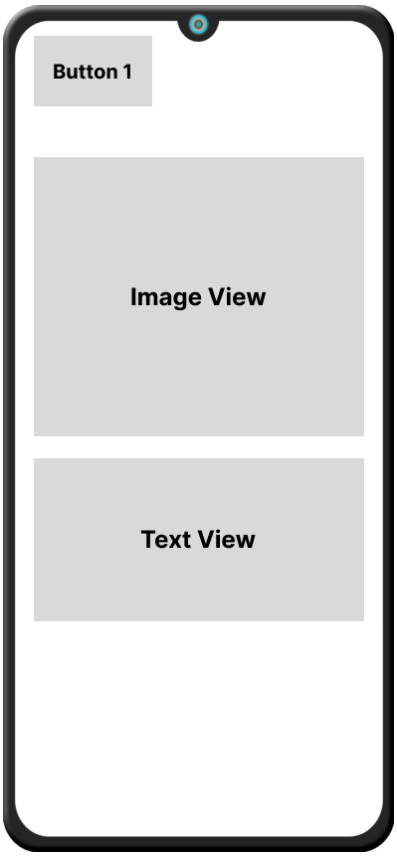
Gambar 3. 31 Antarmuka Menu Utama

Berdasarkan gambar 3.31 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Image View	Menampilkan gambar tomat
2	Text View	Menampilkan nama aplikasi
3	Button 1	Tombol menu bantuan
4	Button 2	Tombol ambil via kamera
5	Button 3	Tombol ambil via galeri
6	Button 4	Tombol deteksi *realtime*
7	Button 5	Tombol navigasi menu utama

8	Button 6	Tombol navigasi menu riwayat
9	Button 7	Tombol navigasi menu tentang

3.4.5.3 Rancangan Antarmuka Menu Hasil Analisa

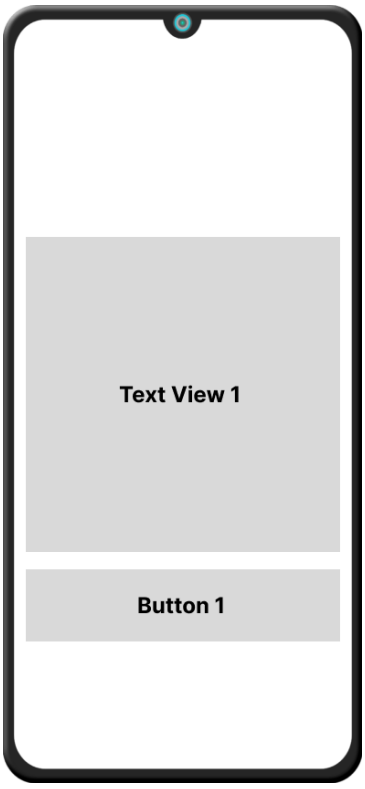


Gambar 3. 32 Antarmuka Menu Hasil Klasifikasi

Berdasarkan gambar 3.32 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Image View	Menampilkan pratinjau gambar
2	Text View	Menampilkan informasi penyakit daun tomat
3	Button 1	Tombol untuk kembali ke halaman sebelumnya

3.4.5.4 Rancangan Antarmuka Notifikasi Gagal Deteksi



Gambar 3. 33 Antarmuka Notifikasi Gagal Deteksi

Berdasarkan gambar 3.33 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

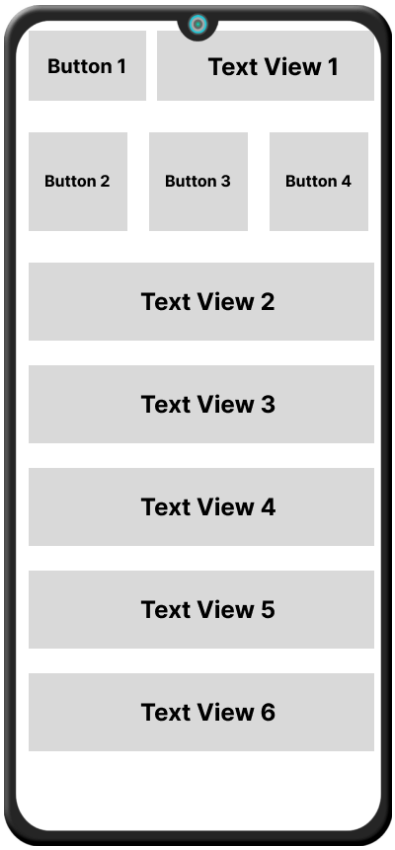
3.4.5.5 Rancangan Antarmuka Menu Deteksi *Realtime*

Gambar 3. 34 Antarmuka Menu Deteksi *Realtime*

Berdasarkan gambar 3.34 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Camera Preview	Menampilkan logo aplikasi
2	Text View 1	Menampilkan informasi hasil deteksi
3	Button 1	Tombol untuk kembali

3.4.5.6 Rancangan Antarmuka Panduan



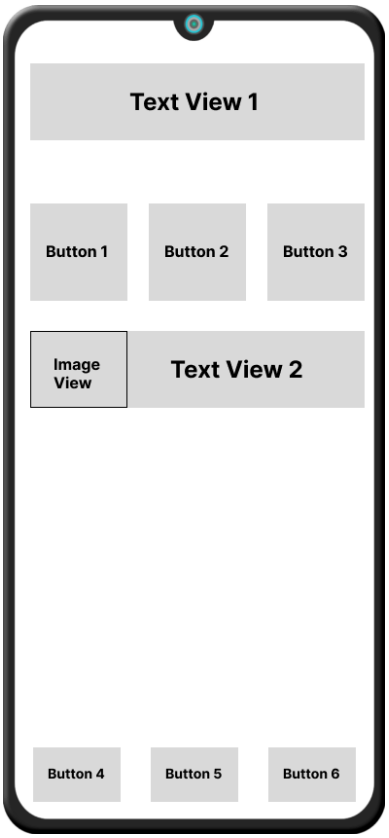
Gambar 3. 35 Antarmuka Panduan

Berdasarkan gambar 3.35 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Text View 1	Menampilkan informasi menu
2	Text View 2	Menampilkan informasi panduan penggunaan
3	Text View 3	Menampilkan informasi panduan penggunaan
4	Text View 4	Menampilkan informasi panduan penggunaan
5	Text View 5	Menampilkan informasi panduan penggunaan

6	Text View 6	Menampilkan informasi panduan penggunaan
7	Button 1	Tombol untuk kembali ke halaman sebelumnya
8	Button 2	Tombol untuk menampilkan panduan ambil via galeri
9	Button 3	Tombol untuk menampilkan panduan ambil via kamera
10	Button 4	Tombol untuk menampilkan panduan ambil deteksi *realtime*

3.4.5.1 Rancangan Antarmuka Riwayat

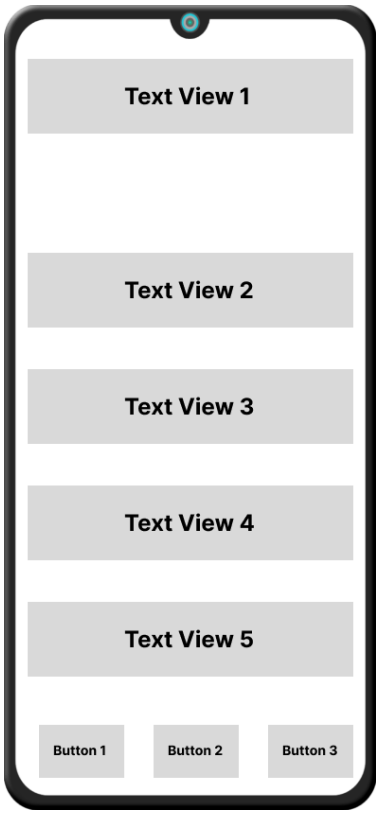


Gambar 3. 36 Antarmuka Riwayat

Berdasarkan gambar 3.36 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Image View	Menampilkan pratinjau riwayat gambar
2	Text View 1	Menampilkan informasi menu
3	Text View 2	Menampilkan riwayat nama penyakit
4	Button 1	Tombol navigasi menu utama
5	Button 2	Tombol navigasi menu riwayat
6	Button 3	Tombol navigasi menu tentang

3.4.5.2 Rancangan Antarmuka Menu Tentang



Gambar 3. 37 Antarmuka Tentang

Berdasarkan gambar 3.37 Pada antarmuka menu utama terdapat beberapa item dan fungsi sebagai berikut:

No	Item	Fungsi
1	Text View 1	Menampilkan nama aplikasi
2	Text View 2	Menampilkan nama pengembang

3	Text View 3	Menampilkan teknologi yang digunakan
4	Text View 4	Menampilkan informasi performa model
5	Text View 5	Menampilkan informasi aplikasi