BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Kerangka Berpikir dan Metode Penelitian

Pada penelitian ini dilakukan pengembangan sistem deteksi penyakit tanaman tomat berbasis aplikasi Android menggunakan teknologi *Deep Learning*. Sistem dirancang untuk membantu pengguna dalam mengidentifikasi penyakit pada daun tomat secara otomatis melalui citra digital yang diambil menggunakan kamera perangkat atau dipilih dari galeri. Model *Deep Learning* yang digunakan adalah *MobileNetV3* yang telah dikonversi ke format *TensorFlow Lite* sehingga dapat dijalankan secara luring (*offline*) pada perangkat Android tanpa memerlukan koneksi internet.

Penelitian ini diawali dengan proses identifikasi permasalahan yang terjadi pada budidaya tanaman tomat, khususnya dalam proses deteksi penyakit yang masih dilakukan secara manual serta adanya penurunan performa model ketika hanya menggunakan dataset laboratorium. Untuk mengatasi permasalahan tersebut, dilakukan pembangunan *Hybrid dataset* yang menggabungkan dataset *PlantVillage* dan dataset citra daun tomat kondisi nyata yang diperoleh dari area pertanian di Pangalengan. Dataset tersebut kemudian digunakan untuk melatih model *MobileNetV3* agar memiliki kemampuan generalisasi yang lebih baik pada kondisi lapangan.

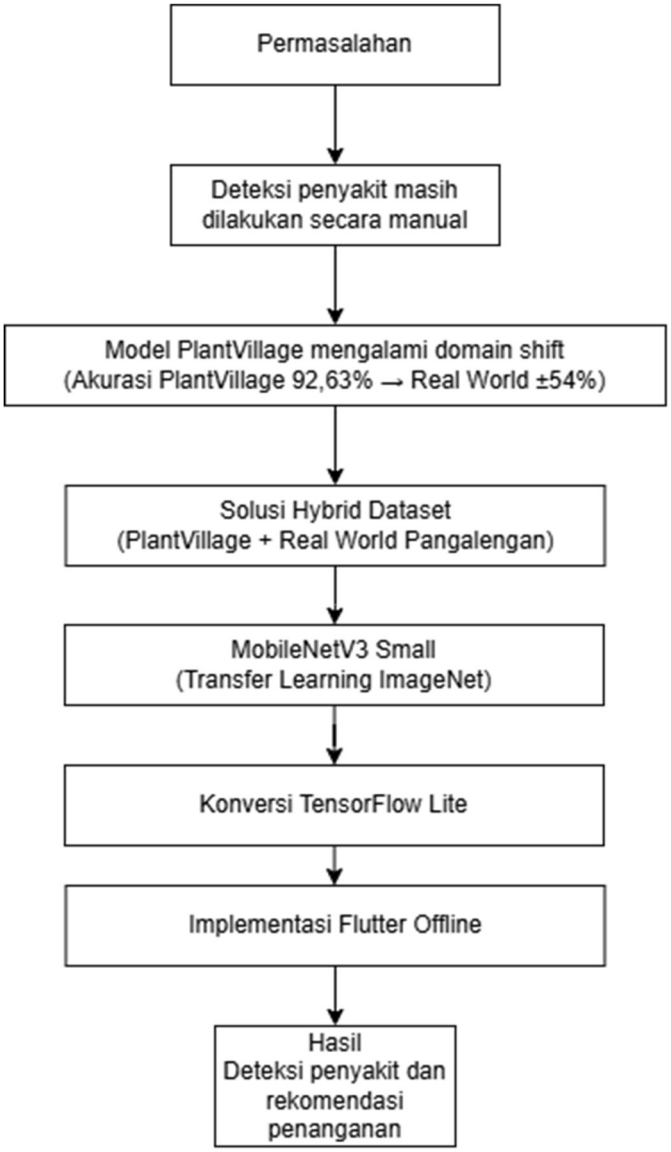
Dalam proses pengembangan sistem, penelitian ini menggunakan metode *Prototyping* yang memungkinkan proses perancangan, evaluasi, dan penyempurnaan aplikasi dilakukan secara bertahap. Model yang telah dilatih kemudian dikonversi ke format *TensorFlow Lite* dan diintegrasikan ke dalam aplikasi Android berbasis *Flutter*. Tahap akhir penelitian dilakukan melalui pengujian model dan pengujian aplikasi untuk mengetahui performa klasifikasi serta fungsionalitas sistem yang dikembangkan.

3.1.1 Kerangka Berpikir

Kerangka berpikir merupakan gambaran alur logis penelitian yang menunjukkan hubungan antara permasalahan, solusi yang diusulkan, proses penelitian, hingga hasil yang diharapkan. Permasalahan utama dalam penelitian ini adalah kesulitan pengguna dalam mengidentifikasi penyakit tanaman tomat secara cepat dan akurat karena keterbatasan pengetahuan serta kurangnya akses terhadap tenaga ahli pertanian.

Untuk mengatasi permasalahan tersebut, penelitian ini memanfaatkan teknologi *Deep Learning* menggunakan arsitektur *MobileNetV3* untuk mengklasifikasikan citra daun tomat ke dalam lima kelas, yaitu *Early Blight*, *Late Blight*, *Leaf Mold*, *Septoria Leaf Spot*, dan *Healthy*. Selain itu, penelitian ini menerapkan pendekatan *Hybrid dataset* yang menggabungkan dataset *PlantVillage* dan dataset citra daun tomat kondisi nyata dari area pertanian di Pangalengan guna meningkatkan kemampuan generalisasi model terhadap kondisi lapangan dan mengurangi pengaruh *Domain shift* antara data pelatihan dan data pengujian.

Model yang telah dilatih kemudian dikonversi ke format *TensorFlow Lite* dan diintegrasikan ke dalam aplikasi Android berbasis *Flutter* sehingga proses deteksi dapat dilakukan secara luring (*offline*) langsung pada perangkat pengguna. Melalui pendekatan tersebut, pengguna diharapkan dapat memperoleh hasil identifikasi penyakit beserta rekomendasi penanganan secara cepat, praktis, dan tanpa ketergantungan terhadap koneksi internet.



Gambar 3. 1 Kerangka Berpikir

Sumber : Penulis (2026)

3.1.2 Metode Penelitian

Penelitian ini mengadopsi pendekatan penelitian terapan (*applied research*). Pemilihan pendekatan tersebut didasarkan pada tujuan akhir studi yang tidak sekadar mengkaji literatur secara teoretis, melainkan berfokus pada penciptaan produk nyata berupa aplikasi *Android* berbasis *Deep Learning* untuk mengidentifikasi penyakit daun tomat secara otonom. Untuk siklus rekayasa perangkat lunaknya, digunakan metode *Prototyping*. Melalui metode ini, tahapan

desain, penilaian, hingga iterasi perbaikan sistem dapat dieksekusi secara gradual dengan berpusat pada spesifikasi kebutuhan pengguna.

Adapun data yang dimanfaatkan mencakup *Hybrid dataset* berjumlah 7.473 gambar daun tomat. Kumpulan data ini merupakan hasil perpaduan antara dataset *PlantVillage* dengan sampel visual dari kondisi lapangan riil yang diambil langsung dari perkebunan tomat di kawasan Pangalengan. Seluruh citra dikategorikan ke dalam lima klasifikasi utama: *Early Blight, Healthy, Late Blight, Leaf Mold, serta Septoria Leaf Spot*. Keseluruhan data ini selanjutnya diproses pada fase pelatihan model *MobileNetV3 Small* dengan memanfaatkan teknik *Transfer Learning* yang menginisialisasi bobot pra-latih dari *ImageNet.*

Secara garis besar, alur kerja diawali oleh tahap akuisisi dan prapemrosesan dataset, yang kemudian diteruskan pada fase training model *Deep Learning* berarsitektur *MobileNetV3 Small*. Guna meninjau tingkat keandalan model dalam mengklasifikasikan jenis penyakit daun tomat, tahap pengujian dilakukan dengan meninjau matriks evaluasi seperti *Accuracy, Precision, Recall, F1-Score ,* beserta *Confusion Matrix*. Apabila hasil kinerja model dirasa sudah optimal, arsitektur tersebut diubah bentuknya menjadi format *TensorFlow Lite* (.tflite) agar siap ditanamkan ke dalam lingkungan aplikasi Android yang dibangun melalui *framework Flutter.*

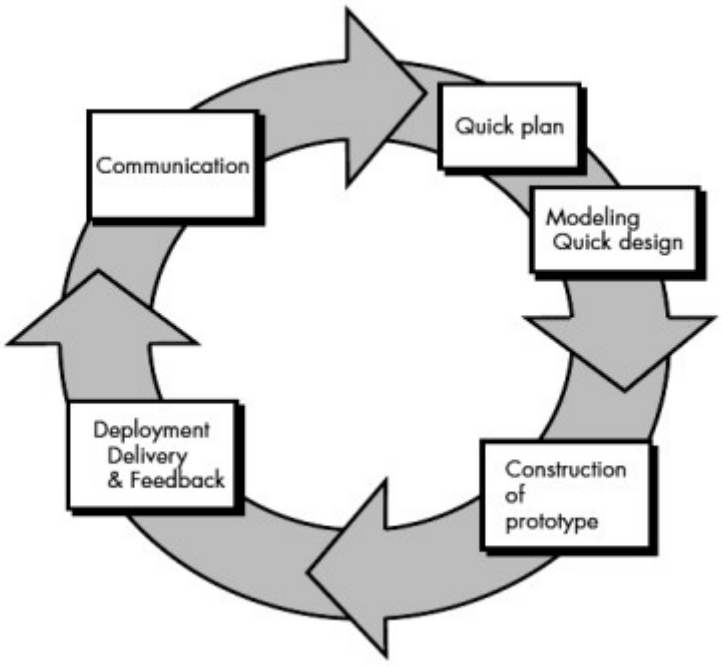
Sebagai langkah penutup, uji fungsionalitas sistem dilaksanakan guna memvalidasi bahwa seluruh komponen aplikasi dapat beroperasi dengan lancar, sekaligus membuktikan kapabilitasnya dalam mengeksekusi prediksi penyakit secara *offline* langsung dari gawai Android. Keluaran akhir dari riset ini ditargetkan mampu memfasilitasi pengguna dalam mendiagnosis masalah kesehatan pada tanaman tomat dengan jauh lebih efisien dan instan, tanpa harus bergantung pada ketersediaan koneksi internet.

3.1.3 Metode Pengembangan Perangkat Lunak

Dalam merancang perangkat lunak untuk penelitian ini, pendekatan utama yang diaplikasikan adalah model *Prototyping*. Alasan mendasar di balik pemilihan

model ini adalah kapabilitasnya dalam memfasilitasi pembangunan sistem secara iteratif (bertahap). Pendekatan ini dimulai dengan menciptakan rancangan prototype dasar, yang selanjutnya akan terus ditinjau serta disempurnakan agar benar-benar selaras dengan spesifikasi kebutuhan yang diharapkan.

Secara prosedural, fase penerapan *Prototyping* di dalam studi ini mencakup serangkaian langkah spesifik, yakni: identifikasi kebutuhan, konstruksi prototype, eksekusi rancang bangun sistem, tahap uji coba, hingga proses evaluasi. Keseluruhan siklus tersebut akan terus dirotasi secara berulang sampai menghasilkan sebuah aplikasi deteksi penyakit tanaman tomat yang beroperasi secara optimal dan sejalan dengan target pencapaian riset.



Gambar 3. 2 Metode Protoyping

Sumber : Diadaptasi dari Bjarnason et al.,(2023)

1. Pada penelitian ini, tahap *Communication* dilakukan dengan mengidentifikasi permasalahan yang menjadi dasar pengembangan aplikasi deteksi penyakit tanaman tomat. Identifikasi dilakukan melalui analisis terhadap permasalahan yang telah diuraikan pada latar belakang penelitian,

yaitu masih sulitnya proses identifikasi penyakit tanaman tomat secara manual karena beberapa jenis penyakit memiliki karakteristik gejala yang hampir serupa. Berdasarkan hasil identifikasi tersebut, dirumuskan kebutuhan sistem berupa aplikasi berbasis Android yang mampu mendeteksi lima jenis kondisi daun tomat menggunakan model *Deep Learning MobileNetV3*, menampilkan hasil deteksi beserta rekomendasi penanganan, serta dapat digunakan secara *offline* melalui implementasi *TensorFlow Lite* .

2. Pada penelitian ini, tahap *Quick Plan* dilakukan dengan menyusun rencana pengembangan sistem berdasarkan hasil identifikasi kebutuhan pada tahap sebelumnya. Perencanaan meliputi penentuan *Hybrid dataset* yang terdiri dari dataset *PlantVillage* dan dataset nyata sebagai data pelatihan model, pemilihan arsitektur *MobileNetV3* sebagai model klasifikasi, penggunaan *Flutter* untuk pengembangan aplikasi Android, serta implementasi *TensorFlow Lite* agar proses deteksi dapat dilakukan secara *offline*. Selain itu, pada tahap ini juga ditentukan kebutuhan perangkat keras dan perangkat lunak yang digunakan selama proses pengembangan sistem.
3. Pada penelitian ini, tahap *Modeling Quick Design* dilakukan dengan merancang struktur dan alur kerja sistem sebelum proses implementasi dimulai. Perancangan dilakukan menggunakan Unified Modeling Language (UML) yang terdiri atas Diagram *Use Case*, Diagram Activity, Diagram Sequence, dan Diagram Class untuk menggambarkan kebutuhan fungsional, alur proses, interaksi antarobjek, serta struktur kelas yang digunakan pada sistem. Selain itu, dilakukan pula perancangan antarmuka (wireframe) sebagai gambaran awal tampilan aplikasi *TomatoCare AI* sehingga proses pengembangan dapat dilakukan secara lebih terstruktur dan sesuai dengan kebutuhan sistem yang telah diidentifikasi.
4. Pada penelitian ini, tahap *Construction of Prototype* merupakan proses penerjemahan hasil perancangan ke dalam bentuk kode program (coding). Pada tahap ini, antarmuka pengguna dibangun menggunakan framework *Flutter* dengan 6ublic pemrograman *Dart* agar aplikasi dapat beroperasi

secara optimal pada sistem operasi Android. Selanjutnya, model *MobileNetV3* yang telah selesai dilatih dan dikonversi ke dalam format *TensorFlow Lite* (.tflite) diintegrasikan langsung ke dalam kerangka kerja aplikasi, sehingga fungsi inferensi deteksi penyakit dapat dieksekusi secara 7ubli di dalam memori perangkat tanpa memerlukan koneksi internet.

5. Pada penelitian ini, tahap *Deployment, Delivery and Feedback* dilakukan dengan membangun aplikasi menjadi file APK kemudian menginstalnya pada perangkat Android untuk dilakukan pengujian fungsional dan pengujian lapangan. Karena aplikasi dirancang bekerja secara *offline* menggunakan *TensorFlow Lite* , proses deployment dilakukan langsung pada perangkat pengguna tanpa memerlukan server. Hasil pengujian tersebut kemudian digunakan sebagai bahan evaluasi untuk memastikan sistem telah berjalan sesuai dengan kebutuhan pengguna.

3.2 Prosedur Penelitian

Prosedur penelitian ini disusun secara sistematis dan terstruktur layaknya sebuah resep panduan instruksional. Pendekatan langkah demi langkah ini diterapkan guna memastikan seluruh tahapan pengembangan sistem deteksi penyakit daun tomat dapat direplikasi secara jelas, runtut, dan terukur. Merujuk pada alur diagram penelitian, eksekusi dilakukan melalui urutan langkah berikut

1. **Tahap Analisis dan Perancangan Sistem**

   Langkah pertama adalah mengidentifikasi masalah utama di lapangan, menentukan kebutuhan sistem secara fungsional maupun *non-fungsional*, serta merancang arsitektur kecerdasan buatan dan basis data (khusus portal distribusi) yang akan digunakan.

2. **Tahap Pengumpulan *Hybrid Dataset***

   Langkah kedua adalah mengakuisisi citra dari dua sumber berbeda, yakni mengunduh dataset laboratorium publik *(PlantVillage)* dan mengambil foto daun tomat kondisi nyata secara langsung di area perkebunan Pangalengan. Kedua data tersebut digabungkan menjadi satu direktori terpadu (*Hybrid dataset).*

3. **Tahap Rancang UI/UX (*User Interface/User Experience*)**

   Langkah ketiga adalah membuat perancangan cetak biru visual *(Modeling Quick Design)* menggunakan *Unified Modeling Language* (UML) dan membangun wireframe antarmuka aplikasi *TomatoCare AI* agar intuitif dan mudah digunakan oleh petani.

4. **Tahap Pelatihan Model *MobileNetV3***

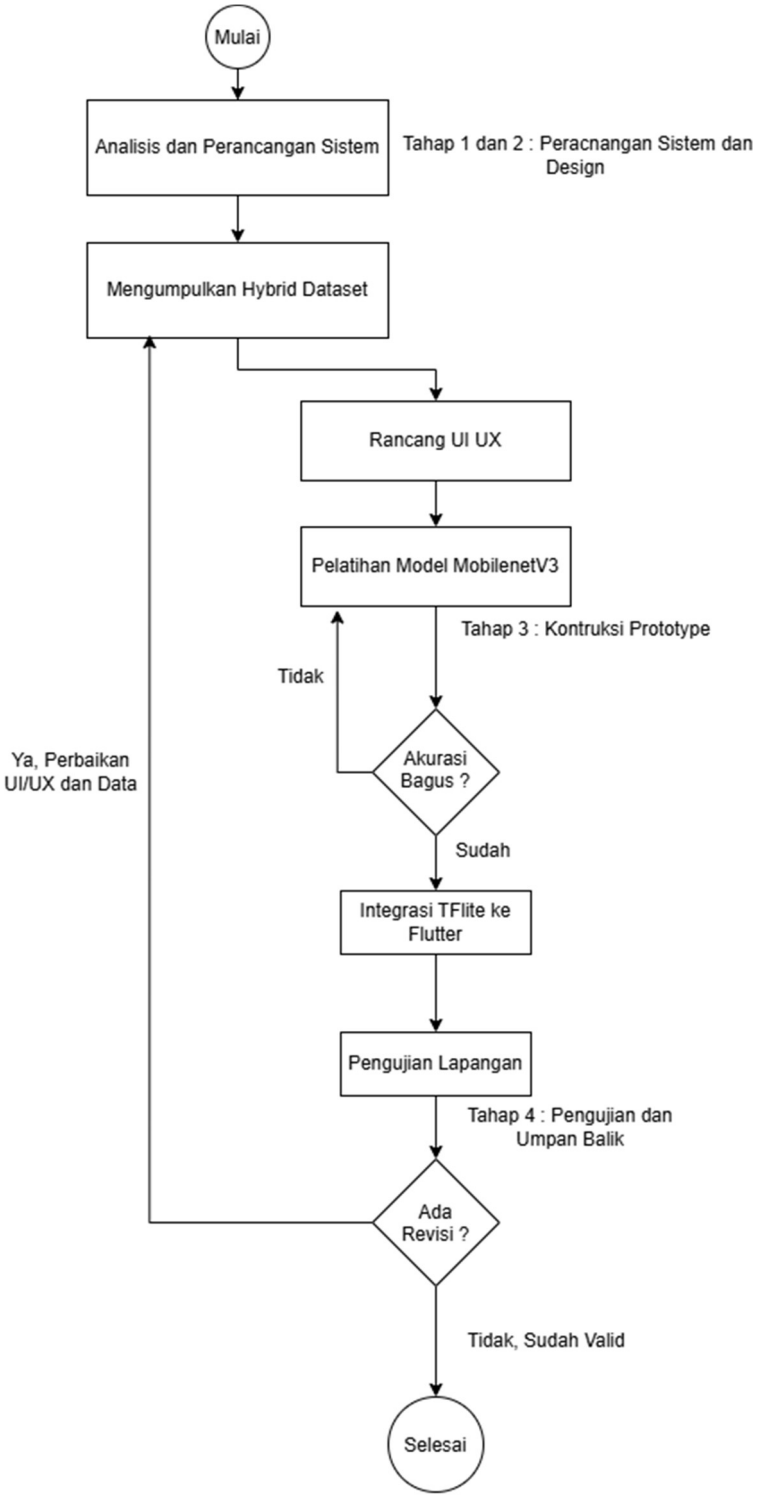
   Langkah keempat adalah melatih algoritma *Deep Learning MobileNetV3* Small menggunakan *Hybrid datase*t yang telah diprapemrosesan. Pada tahap ini terdapat proses evaluasi; apabila akurasi dirasa belum bagus, maka akan dilakukan *looping* (pengulangan) pelatihan model hingga mencapai tingkat akurasi yang optimal.

5. **Tahap Integrasi *TFLite* ke *Flutter***

   Langkah kelima adalah mengonversi model dengan performa terbaik ke dalam format *TensorFlow Lite* (.tflite) agar komputasinya menjadi ringan. Model tersebut kemudian ditanamkan (*embedded*) ke dalam basis kode framework *Flutter* sehingga aplikasi Android dapat beroperasi sepenuhnya secara luring *(offline*).

6. **Tahap Pengujian Lapangan & Umpan Balik:**

   Langkah terakhir adalah membawa aplikasi yang sudah dibangun ke area perkebunan untuk diuji coba langsung oleh pengguna. Apabila terdapat feedback atau revisi, maka akan dilakukan perbaikan pada UI/UX maupun penambahan data. Jika hasil pengujian sudah divalidasi dan berjalan lancar, maka prosedur penelitian dinyatakan selesai.



Gambar 3. 3 Diagram Prosedur Penelitian

Sumber : Penulis (2026)

3.2.1 Tahapan Pengumpulan Data

Tahap pengumpulan data dilakukan dengan mengumpulkan citra daun tomat yang berasal dari dua sumber data, yaitu dataset *PlantVillage* dan dataset kondisi nyata (real-world dataset). Penggunaan dua sumber data ini bertujuan untuk meningkatkan keragaman karakteristik citra sehingga model dapat mempelajari kondisi daun tomat baik pada lingkungan laboratorium maupun kondisi nyata di lapangan, serta meningkatkan kemampuan generalisasi model terhadap data baru.

Untuk membentuk *Hybrid dataset* yang komprehensif, tahapan pengumpulan dan penggabungan dilakukan melalui rincian berikut:

1. **Pengumpulan Dataset Laboratorium (*PlantVillage*)**

   Dataset *PlantVillage* digunakan sebagai sumber data laboratorium yang telah memiliki label untuk setiap kelas penyakit. Dataset ini diunduh melalui repositori 10ublic dengan mengambil lima kelas kondisi daun, yaitu *Healthy*, *Early Blight*, *Late Blight*, *Leaf Mold*, dan *Septoria Leaf Spot*. Tautan (URL) unduhan dataset sumber dari repositori Kaggle ini dapat dilihat secara lengkap pada Lampiran 22.

2. **Pengumpulan Dataset Kondisi Nyata (Real-World)**

   Dataset kondisi nyata diperoleh melalui proses pengambilan citra secara langsung pada tempat budidaya tanaman tomat yang berlokasi di Kampung Cilaku, Desa Sukaluyu, Kecamatan Pangalengan, Kabupaten Bandung, Jawa Barat. Pengambilan citra dilakukan pada berbagai waktu untuk mendapatkan variasi pencahayaan dan gangguan visual (noise) alami. Seluruh citra mentah hasil akuisisi lapangan ini diunggah ke penyimpanan Google Drive, yang tautan aksesnya juga disertakan pada Lampiran 22.

3. **Pembentukan Hybrid Dataset**

   Setelah kedua kelompok data terkumpul, dilakukan penyaringan (data cleaning) untuk membuang citra nyata yang tidak terfokus. Citra yang lolos seleksi kemudian digabungkan (*merging*) dengan dataset *PlantVillage* ke dalam satu direktori terpadu yang disebut *Hybrid dataset* dengan total

keseluruhan mencapai 7.473 citra. Seluruh dataset gabungan ini disimpan ke dalam media penyimpanan cloud untuk kebutuhan prapemrosesan dan pelatihan model. Tautan akses menuju direktori Google Drive yang memuat keseluruhan *Hybrid dataset* siap latih ini dapat ditinjau pada Lampiran 22.

Tabel 3. 1 Distribusi *Hybrid dataset*

No	Kelas	*PlantVillage*	Real World	*Hybrid dataset*
1	*Early Blight*	1000	42	1042
2	*Healthy*	1591	100	1691
3	*Late Blight*	1909	16	1925
4	*Leaf Mold*	952	0	952
5	*Septoria Leaf Spot*	1771	92	1863
	Jumlah			7473

Sumber: Penulis (2026)

3.2.2 Tahapan Pelatihan Model

Fase training (pelatihan) komputasi baru dieksekusi pasca-rampungnya proses akuisisi, seleksi, serta kategorisasi seluruh sampel citra ke dalam masing-masing kelas penyakit. *Hybrid dataset* yang telah terhimpun ini selanjutnya dipartisi ke dalam tiga himpunan distribusi, yakni data latih (*training*), data validasi *(validation),* serta data uji *(testing*). Pemecahan komposisi data ini diimplementasikan guna memfasilitasi tahapan konstruksi sekaligus penilaian kinerja model secara lebih empiris dan objektif.

Sebagai pra-syarat sebelum model mulai dilatih, seluruh sampel gambar wajib melewati tahapan *preprocessing* (prapemrosesan). Rangkaian langkah ini mencakup penyesuaian dimensi resolusi (*resizing*) ke ukuran 224×224 piksel, yang dibarengi dengan proses normalisasi nilai piksel ke dalam skala 0 hingga 1. Guna meminimalisir potensi terjadinya *overfitting* sekaligus memperkaya keberagaman sampel pada data latih, sistem juga menerapkan fungsi *Data Augmentation* yang

terdiri atas manipulasi RandomFlip, RandomRotation, *RandomZoom*, *RandomContrast*, hingga *RandomTranslation*.untuk meningkatkan variasi data pelatihan serta mengurangi risiko *overfitting.*

Untuk arsitektur utamanya, fase training ini mengadopsi *MobileNetV3 Small* yang dioptimasi melalui mekanisme *Transfer Learning* berbasis bobot bawaan dari *ImageNet*. Begitu siklus pelatihan tuntas, unjuk kerja model akan langsung divalidasi menggunakan himpunan data validasi. Tolok ukur kemampuannya dalam mengidentifikasi jenis penyakit pada daun tomat dianalisis dengan bersandar pada parameter *Accuracy, Precision, Recall, F1-Score* , beserta *Confusion Matrix.* Apabila ditemukan model dengan catatan performa paling unggul, struktur tersebut akan diekspor dan dikonversi wujudnya ke dalam ekstensi *TensorFlow Lite* (.tflite) supaya kompatibel saat diintegrasikan ke dalam lingkungan aplikasi Android yang dibangun dengan kerangka kerja *Flutter*.

Tabel 3. 2 Pembagian Dataset

No	Kelas	Training	Validation	Testing
1	*Early Blight*	729	156	157
2	*Healthy*	1183	254	254
3	*Late Blight*	1347	289	289
4	*Leaf Mold*	666	143	143
5	*Septoria Leaf Spot*	1304	279	280
	Total	5229	1121	1123

3.2.3 Tahapan Implementasi Sistem

Tahap implementasi sistem dilakukan setelah model *MobileNetV3 Small* selesai dilatih dan memperoleh performa terbaik berdasarkan hasil evaluasi. Model yang telah disimpan kemudian dikonversi ke format *TensorFlow Lite* (.tflite) agar dapat dijalankan pada perangkat Android dengan kebutuhan sumber daya yang lebih ringan.

Selanjutnya, model *TensorFlow Lite* diintegrasikan ke dalam aplikasi Android yang dikembangkan menggunakan *framework Flutter*. Aplikasi dirancang untuk memungkinkan pengguna mengambil gambar daun tomat melalui kamera atau memilih gambar dari galeri perangkat sebagai masukan untuk proses deteksi penyakit.

Pada tahap ini juga dilakukan implementasi proses prapemrosesan citra, proses inferensi menggunakan model *TensorFlow Lite* , serta penyajian hasil klasifikasi kepada pengguna. Hasil yang ditampilkan meliputi jenis penyakit yang terdeteksi, nilai *confidence score*, dan rekomendasi penanganan penyakit. Seluruh proses deteksi dilakukan secara luring (*offline*) pada perangkat tanpa memerlukan koneksi internet maupun server eksternal.

3.2.4 Tahapan Pengujian

Tahap pengujian dilakukan untuk mengetahui kinerja model serta memastikan seluruh fungsi aplikasi berjalan sesuai dengan kebutuhan penelitian. Pengujian model dilakukan menggunakan data uji (testing data) yang tidak digunakan selama proses pelatihan maupun validasi sehingga hasil evaluasi dapat menggambarkan kemampuan model dalam melakukan klasifikasi penyakit daun tomat secara objektif.

Evaluasi model dilakukan menggunakan metrik *Accuracy*, *Precision*, *Recall*, *F1-Score* , dan *Confusion Matrix* untuk mengukur performa klasifikasi pada setiap kelas penyakit. Selain itu, dilakukan pengujian perangkat lunak menggunakan metode *Black Box* Testing untuk memastikan seluruh fitur aplikasi, seperti pemilihan gambar, proses deteksi penyakit, perhitungan *confidence score*, dan penyajian rekomendasi penanganan dapat berjalan sesuai dengan fungsinya. Hasil pengujian digunakan sebagai dasar dalam menilai performa model serta keandalan sistem yang dikembangkan.

3.3 Analisis Sistem

Analisis sistem merupakan tahapan yang dilakukan untuk memahami permasalahan yang melatarbelakangi pengembangan sistem deteksi penyakit daun tomat serta menentukan kebutuhan yang diperlukan dalam proses pembangunan

aplikasi. Pada tahap ini dilakukan analisis terhadap proses identifikasi penyakit yang masih dilakukan secara manual, keterbatasan penggunaan model yang hanya dilatih menggunakan dataset laboratorium, serta kebutuhan pengguna terhadap sistem deteksi penyakit yang dapat beroperasi secara cepat dan *offline* pada perangkat bergerak.

Selain itu, dilakukan analisis terhadap sistem yang diusulkan, kebutuhan perangkat lunak dan perangkat keras, serta alur proses deteksi penyakit menggunakan model *Deep Learning MobileNetV3 Small* yang diintegrasikan ke dalam aplikasi Android berbasis *Flutter*. Hasil analisis ini menjadi dasar dalam proses perancangan dan implementasi sistem pada tahapan berikutnya.

3.3.1 Gambaran Umum Objek Penelitian

Objek utama dalam penelitian ini adalah citra visual dari daun tanaman tomat (*Solanum lycopersicum*). Tomat merupakan salah satu komoditas hortikultura bernilai ekonomi tinggi di Indonesia, namun memiliki tingkat kerentanan yang sangat signifikan terhadap serangan patogen, seperti jamur, bakteri, dan virus. Daun dipilih sebagai objek observasi yang krusial karena organ vegetatif ini merupakan indikator visual pertama yang merepresentasikan gejala klinis saat tanaman terinfeksi penyakit. Gejala tersebut umumnya bermanifestasi dalam bentuk perubahan morfologi dan pigmen, seperti munculnya bercak (*lesi*), penguningan (*klorosis*), hingga kematian jaringan (*nekrosis*). Oleh karena itu, pemantauan kondisi daun secara presisi menjadi kunci utama dalam upaya deteksi dini, pengendalian wabah, dan penyelamatan kualitas hasil panen.

Dalam penelitian ini, citra daun tomat yang digunakan terdiri atas lima kelas kondisi, yaitu *Healthy, Early Blight, Late Blight, Leaf Mold*, dan *Septoria Leaf Spot* yang menjadi target klasifikasi *model Deep Learning MobileNetV3 Small*. Dataset yang digunakan merupakan *Hybrid dataset*, yaitu gabungan antara dataset *PlantVillage* dan citra daun tomat kondisi nyata (real-world dataset). Pengambilan citra kondisi nyata dilakukan secara langsung di tempat budidaya pribadi tanaman tomat yang berlokasi di Kampung Cilaku, Desa Sukaluyu, Kecamatan Pangalengan, Kabupaten Bandung, Jawa Barat. Lokasi tersebut dipilih karena

memiliki kondisi budidaya tanaman tomat yang representatif sehingga mampu menyediakan variasi citra daun sehat maupun daun yang terinfeksi penyakit untuk mendukung proses pelatihan dan evaluasi model.

3.3.2 Analisis Masalah

Berdasarkan observasi terhadap praktik pemantauan penyakit tanaman yang berjalan saat ini, terdapat beberapa permasalahan fundamental yang dihadapi oleh para pembudidaya tomat. Pertama, proses identifikasi penyakit umumnya masih dilakukan secara konvensional melalui pengamatan visual kasatmata. Pendekatan ini sangat bergantung pada tingkat pengalaman empiris petani, rentan terhadap bias subjektivitas, dan berisiko tinggi memicu kesalahan diagnosis yang berujung pada pemberian pestisida yang tidak tepat sasaran.

Kedua, rasio ketersediaan pakar patologi tumbuhan atau penyuluh pertanian lapangan sangat terbatas jika dibandingkan dengan luasnya area perkebunan. Hal ini mengakibatkan proses konsultasi menjadi lambat dan penanganan penyakit seringkali tertunda hingga infeksi menyebar luas. Ketiga, sebagian besar area lahan pertanian tomat berlokasi di dataran tinggi atau wilayah pedesaan yang memiliki keterbatasan infrastruktur jaringan internet (*blank spot*). Kondisi geografis ini menyebabkan adopsi teknologi pertanian cerdas berbasis komputasi awan (*cloud-based AI*) yang sangat bergantung pada koneksi internet stabil menjadi tidak relevan dan gagal diimplementasikan secara optimal di lapangan.

Selain permasalahan yang berkaitan dengan proses identifikasi penyakit di lapangan, penelitian ini juga menemukan kendala pada tahap pengembangan model klasifikasi. Model awal yang dilatih menggunakan dataset *PlantVillage* mampu menghasilkan akurasi yang tinggi pada data dengan karakteristik serupa, namun mengalami penurunan performa yang signifikan ketika diuji menggunakan citra daun tomat kondisi nyata. Perbedaan karakteristik antara citra laboratorium dan citra lapangan menyebabkan terjadinya *Domain shift* sehingga kemampuan generalisasi model menjadi kurang optimal. Oleh karena itu, diperlukan pendekatan yang mampu meningkatkan keberagaman data pelatihan agar model dapat beradaptasi dengan kondisi lingkungan yang lebih bervariasi.

3.3.3 Analisis Sistem yang Diusulkan

Untuk menjawab berbagai kendala yang telah diuraikan pada subbab sebelumnya, penelitian ini mengusulkan sebuah aplikasi berbasis Android yang dibangun menggunakan framework *Flutter* dan mengandalkan teknologi *Deep Learning* untuk mendeteksi penyakit pada daun tomat secara otomatis. Inti dari sistem ini adalah model *MobileNetV3 Small*, yang proses pelatihannya memanfaatkan *Hybrid dataset* gabungan antara data *PlantVillage* dengan citra kondisi nyata yang diambil langsung dari lahan pertanian tomat di Pangalengan. Setelah proses pelatihan selesai, model tersebut dikonversi ke format *TensorFlow Lite* agar dapat berjalan langsung di perangkat Android.

Salah satu nilai tambah utama dari rancangan sistem ini adalah kemampuannya melakukan *On-Device Inference*, di mana seluruh tahapan klasifikasi citra berlangsung sepenuhnya di perangkat pengguna tanpa membutuhkan sambungan internet ataupun server tambahan. Untuk menggunakannya, pengguna cukup mengambil foto daun tomat langsung melalui kamera atau memilih gambar yang tersimpan di galeri, kemudian model *MobileNetV3 Small* akan memproses citra tersebut secara otomatis.

Setiap hasil klasifikasi yang ditampilkan mencakup jenis penyakit yang terdeteksi, skor keyakinan *(confidence score)*, serta saran penanganan yang relevan. Melalui pendekatan tersebut, aplikasi ini diharapkan dapat menjadi solusi praktis bagi pengguna untuk memperoleh informasi mengenai kondisi tanaman tomat secara cepat, bahkan di wilayah pertanian yang memiliki keterbatasan akses jaringan internet.

3.3.4 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk mendefinisikan spesifikasi teknis yang harus dipenuhi agar sistem dapat dibangun dan dioperasikan sesuai dengan tujuan perancangan. Analisis ini dibagi menjadi empat aspek, yaitu kebutuhan fungsional, non-fungsional, perangkat keras, dan perangkat lunak.

3.3.4.1 Kebutuhan Fungsional

Kebutuhan fungsional mendeskripsikan layanan, proses, atau fitur yang wajib disediakan oleh sistem. Kebutuhan fungsional pada aplikasi ini meliputi:

1. Sistem harus menyediakan fitur integrasi kamera agar pengguna dapat mengambil citra daun tomat secara langsung.
2. Sistem harus menyediakan fitur akses galeri agar pengguna dapat mengunggah citra daun tomat yang telah tersimpan di memori perangkat.
3. Sistem harus mampu mengeksekusi model klasifikasi AI untuk menganalisis citra yang dimasukkan.
4. Sistem harus mampu menampilkan hasil deteksi berupa nama klasifikasi penyakit atau status daun sehat beserta tingkat probabilitas keakuratannya (*confidence score*).
5. Sistem harus menampilkan deskripsi penyakit dan rekomendasi langkah penanganan atau pengobatan yang relevan dengan hasil deteksi.

3.3.4.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mendeskripsikan kriteria operasional dan batasan kualitas dari sistem yang dibangun, yang meliputi:

1. Ketersediaan (*Availability)*: Seluruh fungsionalitas utama aplikasi, khususnya fitur deteksi AI, harus dapat beroperasi 100% tanpa memerlukan koneksi internet.
2. Kinerja *(Performance*): Proses inferensi gambar pada perangkat *full offline* harus berlangsung dengan latensi rendah (cepat) untuk memastikan kenyamanan pengguna.
3. Kebergunaan (*Usability*): Antarmuka pengguna (*User Interface*) harus dirancang secara intuitif, sederhana, dan responsif agar mudah dioperasikan oleh kalangan petani atau pengguna awam yang tidak memiliki latar belakang teknis.

3.3.4.3 Kebutuhan Perangkat Keras *(Hardware*)

Kebutuhan perangkat keras dibagi menjadi dua lingkungan, yaitu lingkungan pengembangan dan lingkungan operasional (pengguna):

1. Perangkat Keras Pengembangan: Komputer/Laptop dengan spesifikasi prosesor multi-inti (*multi-core*), RAM memadai (minimal 8 GB), dan (opsional) Unit Pemrosesan Grafis (GPU) untuk mengakselerasi proses pelatihan model AI.
2. Perangkat Keras Pengguna (Implementasi): *Smartphone* bersistem operasi Android yang dilengkapi dengan modul kamera fungsional dan kapasitas memori internal yang cukup untuk menyimpan ukuran model TFLite.

3.3.4.4 Kebutuhan Perangkat Lunak (*Software*)

Perangkat lunak yang dibutuhkan untuk menunjang proses rekayasa dan pengembangan sistem ini meliputi:

1. Pengembangan Model AI: Bahasa pemrograman Python, library komputasi kecerdasan buatan (*TensorFlow* dan Keras), serta platform Google *Colaboratory* untuk proses pelatihan (training).
2. Pengembangan Aplikasi: *Flutter* SDK, bahasa pemrograman *Dart*, serta *Integrated Development Environment* (IDE) seperti *Visual Studio Code* atau *Android Studio.*

3.3.4.5 Kebutuhan Penyimpanan Data

Pada penelitian ini, aplikasi *TomatoCare AI* tidak menggunakan sistem manajemen basis data (*Database Management System/DBMS*) seperti *MySQL* maupun *Firebase* karena seluruh proses aplikasi dirancang untuk berjalan secara *offline*. Sebagai media penyimpanan data, aplikasi memanfaatkan *SharedPreferences* yang tersedia pada framework *Flutter* untuk menyimpan informasi secara lokal di dalam perangkat Android. Pendekatan ini dipilih karena sesuai dengan kebutuhan aplikasi yang hanya memerlukan penyimpanan data sederhana tanpa memerlukan server maupun koneksi internet.

Data yang disimpan menggunakan *SharedPreferences* berupa riwayat hasil deteksi penyakit tanaman tomat yang meliputi nama penyakit hasil klasifikasi, nilai confidence, waktu deteksi, serta informasi pendukung lainnya yang diperlukan untuk ditampilkan kembali pada halaman History. Dengan mekanisme tersebut, pengguna dapat melihat kembali hasil deteksi sebelumnya meskipun aplikasi telah

ditutup atau perangkat berada dalam kondisi tanpa koneksi internet. Pemanfaatan penyimpanan lokal ini juga mendukung konsep *On-Device Inference*, di mana seluruh proses deteksi dan penyimpanan data dilakukan secara langsung pada perangkat pengguna.

Tabel 3. 3 Kebutuhan Penyimpanan Data

No	Data yang tersimpan	Media Penyimpanan	Keterangan
1	Nama Penyakit	*SharedPreferences*	Menyimpan hasil klasifikasi penyakit
2	Confidence Score	*SharedPreferences*	Menyimpan tingkat keyakinan model
3	Waktu Deteksi	*SharedPreferences*	Menyimpan tanggal dan waktu deteksi
4	Riwayat Deteksi	*SharedPreferences*	Ditampilkan kembali pada halaman History

Sumber : Pemulis (2026)

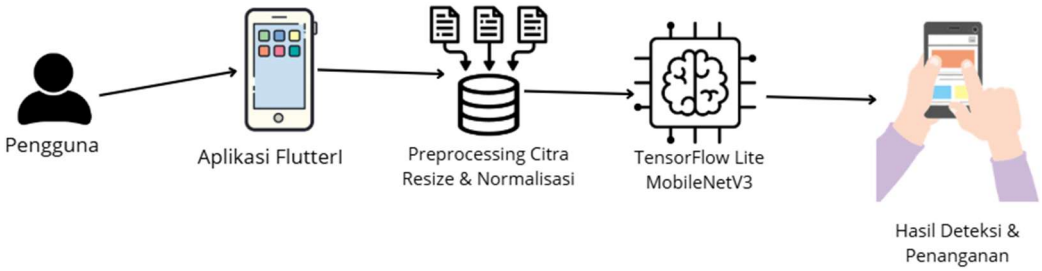
Berdasarkan Tabel 3.3, seluruh data hasil deteksi disimpan menggunakan mekanisme penyimpanan lokal melalui *SharedPreferences*. Pendekatan ini dipilih karena mampu memenuhi kebutuhan aplikasi yang beroperasi secara *offline* dengan proses penyimpanan yang ringan, cepat, serta tidak bergantung pada server eksternal.

3.4 Rancangan Sistem

Setelah kebutuhan sistem dianalisis pada subbab sebelumnya, tahap berikutnya adalah menerjemahkan hasil tersebut ke dalam bentuk rancangan teknis yang akan menjadi acuan dalam membangun aplikasi. Rancangan yang disusun pada tahap ini mencakup empat aspek utama, yaitu arsitektur sistem deteksi penyakit tanaman tomat berbasis *MobileNetV3* Small, alur proses klasifikasi citra, pemodelan sistem menggunakan Unified Modeling Language (UML), serta desain antarmuka pengguna (*User Interface*) untuk aplikasi Android berbasis *Flutter*. Keempat rancangan tersebut selanjutnya menjadi dasar bagi proses implementasi dan pengujian sistem pada tahap berikutnya.

3.4.1 Arsitektur Sistem

Arsitektur sistem pada penelitian ini menggambarkan bagaimana setiap komponen saling terhubung dalam mendukung proses deteksi penyakit tanaman tomat. Alur dimulai ketika pengguna memasukkan citra daun tomat, baik melalui kamera maupun galeri, ke dalam aplikasi Android. Citra tersebut kemudian melewati tahap praproses (*preprocessing*) yang meliputi penyesuaian ukuran gambar dan normalisasi nilai piksel, sebelum akhirnya diklasifikasikan oleh model *MobileNetV3* yang telah dikonversi ke format *TensorFlow Lite* . Keluaran dari proses ini berupa jenis penyakit yang teridentifikasi, tingkat keyakinan prediksi, serta rekomendasi penanganan yang ditampilkan kepada pengguna. Karena seluruh rangkaian proses tersebut berjalan secara lokal (*offline*) di perangkat Android, aplikasi tidak memerlukan koneksi internet sama sekali. Gambaran menyeluruh arsitektur ini dapat dilihat pada Gambar 3.4.



Gambar 3. 4 Arsitektur Sistem

Sumber : Penulis (2026)

3.4.2 Pemodelan Sistem (UML)

Untuk menggambarkan secara lebih rinci bagaimana pengguna berinteraksi dengan aplikasi serta bagaimana alur proses berjalan di dalam sistem, penelitian ini menggunakan pendekatan pemodelan Unified Modeling Language (UML). Pemodelan yang digunakan terdiri atas *Use Case Diagram* dan Activity Diagram, yang masing-masing dijelaskan pada subbab berikut.

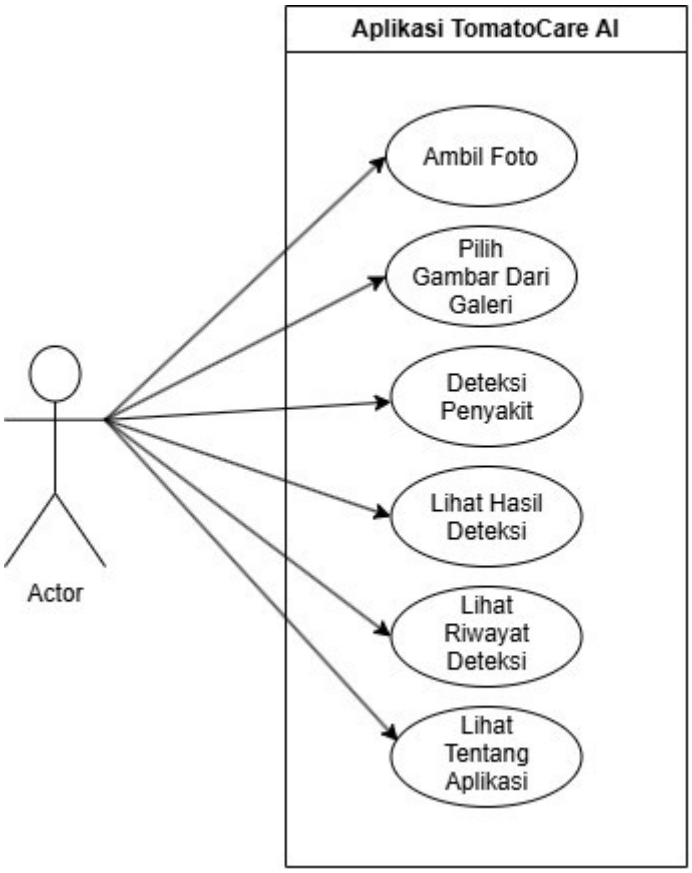
3.4.2.1 *Use Case Diagram*

Use Case Diagram berfungsi untuk memetakan bentuk-bentuk interaksi fungsional yang dapat dilakukan oleh para aktor terhadap sistem yang dibangun. Pada penelitian ini, ruang lingkup interaksi dibagi menjadi dua lingkungan utama, yaitu Aplikasi *Mobile* (luring) dan Portal Web Distribusi (daring).

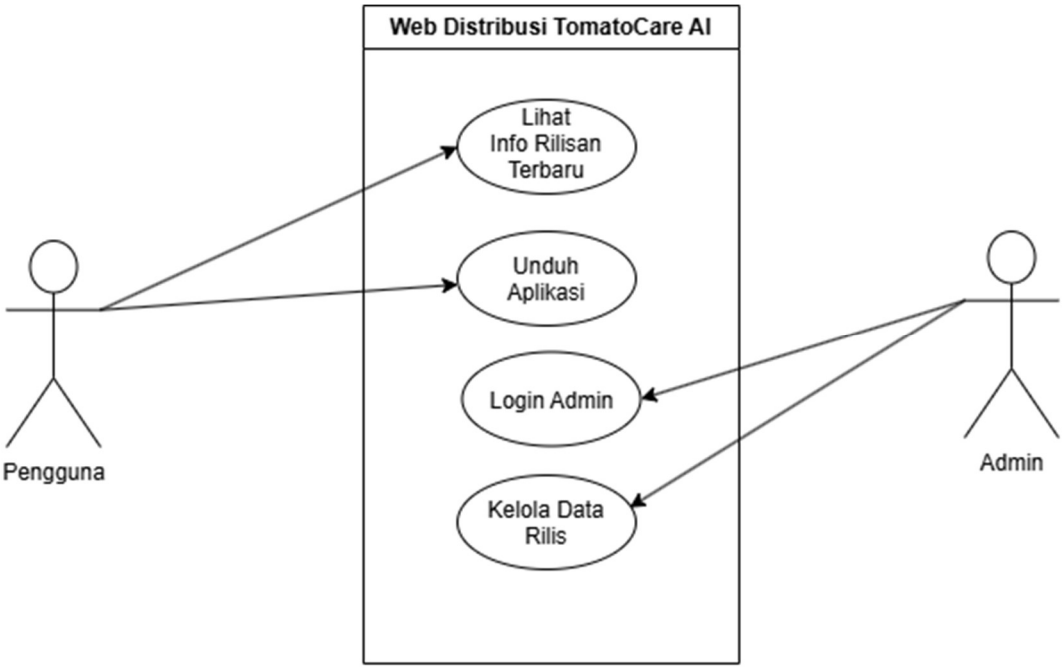
Pada lingkungan Aplikasi *Mobile*, interaksi difokuskan pada aktor Pengguna. Interaksi tersebut dimulai ketika pengguna memilih untuk mengambil foto secara langsung melalui kamera perangkat atau memilih gambar yang sudah tersimpan di galeri. Citra yang dipilih tersebut kemudian akan diproses oleh sistem untuk mendeteksi jenis penyakit pada tanaman tomat, dan hasilnya ditampilkan kembali kepada pengguna beserta rekomendasi penanganan yang sesuai dengan penyakit yang teridentifikasi. Selain itu, pengguna juga dapat berinteraksi dengan sistem untuk meninjau kembali riwayat deteksi yang telah dilakukan

Pada lingkungan Portal Web Distribusi, interaksi melibatkan dua aktor fungsional, yaitu Pengguna dan Admin. Aktor Pengguna dapat mengakses portal secara publik untuk melihat informasi rilis versi terbaru dan mengunduh berkas instalasi aplikasi (.APK). Sementara itu, aktor Admin berinteraksi dengan sistem melalui proses autentikasi (login) untuk mengakses dasbor manajemen. Melalui dasbor tersebut, Admin memiliki otorisasi untuk mengelola, memperbarui, dan

mempublikasikan data rilis aplikasi ke dalam basis data agar dapat diunduh oleh pengguna.



Gambar 3. 5 *Use Case Diagram* Aplikas TomatoCar AI



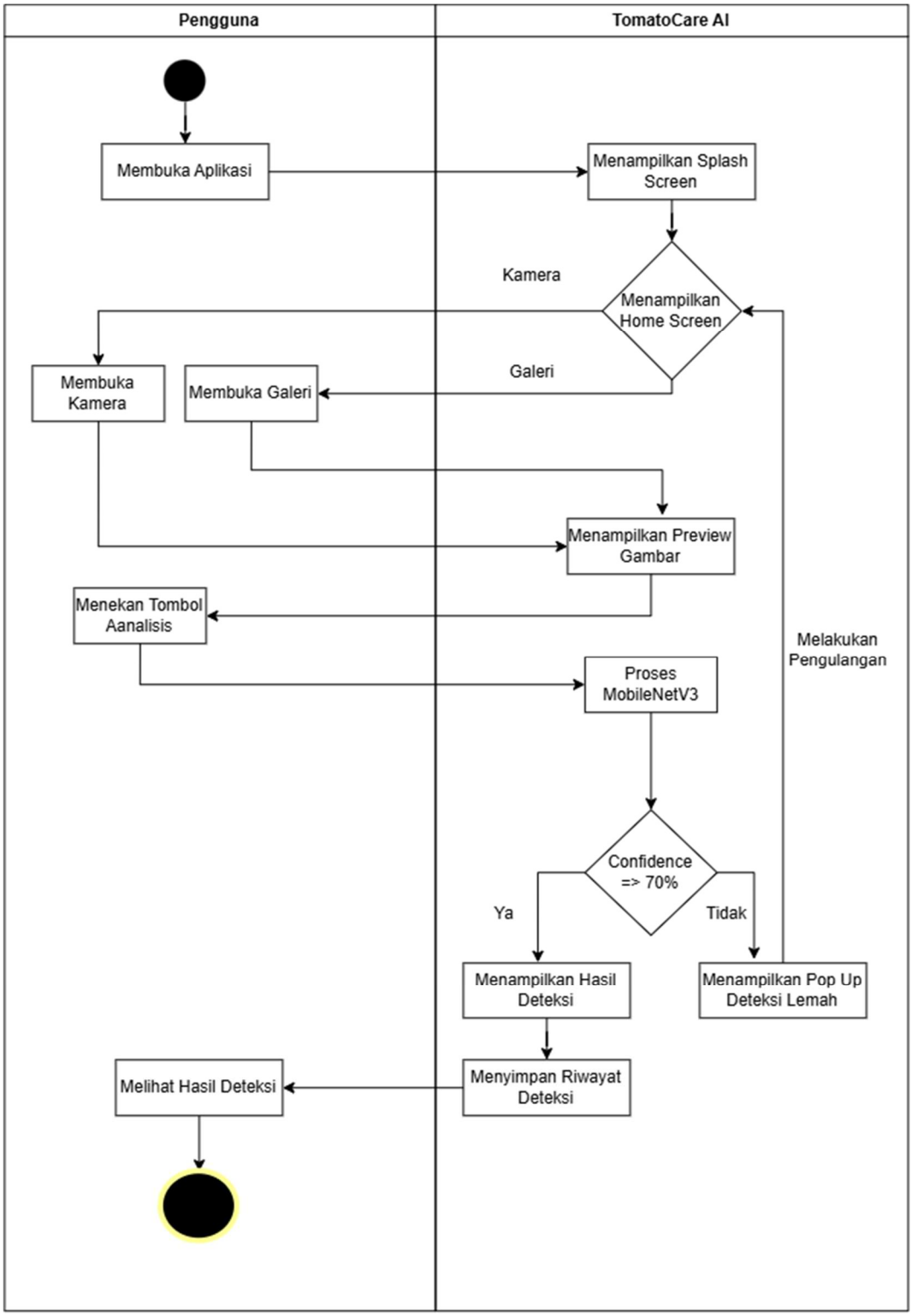
Gambar 3. 6 Diagram Web Distribusi *TomatoCare AI*

3.4.2.2 *Activity Diagram*

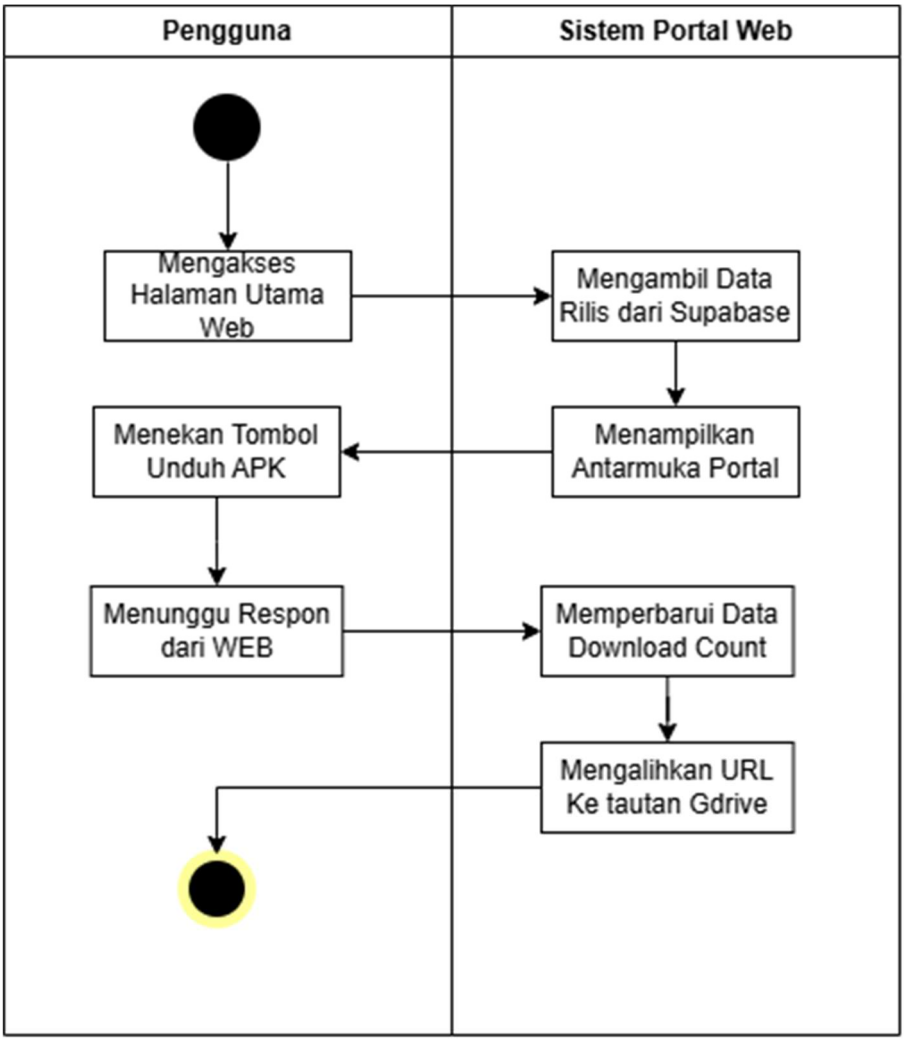
Activity Diagram mendeskripsikan alur kerja dinamis dari berbagai aktivitas operasional yang melibatkan interaksi aktor dengan sistem. Pada perancangan sistem ini, alur aktivitas dibagi menjadi dua bagian, yaitu alur pada Aplikasi *Mobile* dan alur pada Portal Web Distribusi.

Pada lingkungan Aplikasi *Mobile*, alur aktivitas difokuskan pada proses deteksi penyakit. Alur dimulai ketika pengguna membuka aplikasi dan memasukkan citra melalui kamera atau galeri. Sistem kemudian memvalidasi apakah citra berhasil dimuat. Jika berhasil, sistem akan menjalankan fungsi inferensi menggunakan model *TensorFlow Lite* di latar belakang (background thread) agar antarmuka tidak mengalami *freeze*. Setelah proses inferensi selesai, sistem akan mengambil data hasil berupa nama penyakit dan nilai confidence, lalu mencocokkannya dengan basis data lokal untuk mengambil informasi penyebab dan teks rekomendasi penanganan, serta mengakhiri alur dengan menampilkannya secara lengkap pada halaman hasil.

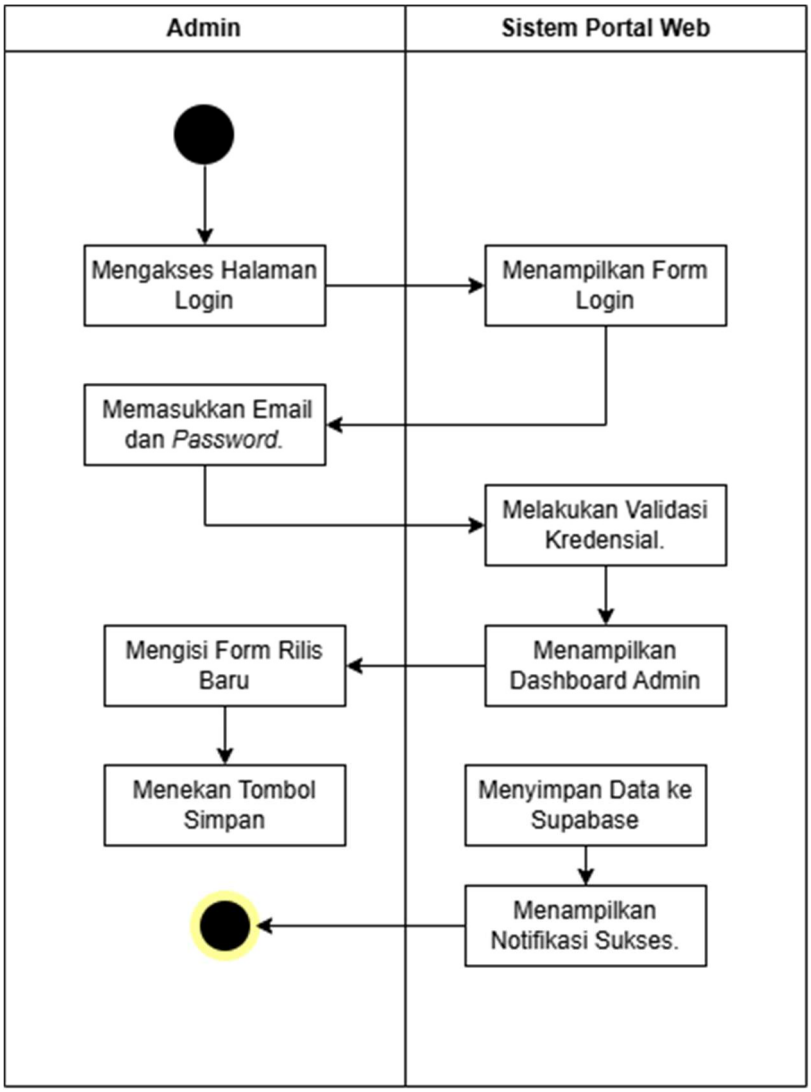
Pada lingkungan Portal Web Distribusi, alur aktivitas dipisahkan berdasarkan peran masing-masing aktor. Alur Pengguna (petani) dimulai dengan mengakses halaman utama, di mana sistem secara otomatis memuat dan menampilkan data rilis aplikasi terbaru. Saat pengguna menekan tombol unduh, sistem memproses penambahan data statistik unduhan secara sistematis di basis data, lalu mengalihkan pengguna ke tautan berkas aplikasi untuk diunduh. Di sisi lain, alur Admin diawali dengan mengakses halaman otorisasi (login). Sistem akan memvalidasi kredensial Admin menggunakan *Supabase* Auth; jika valid, Admin akan diarahkan ke dasbor manajemen. Pada dashboard tersebut, Admin dapat mengisi formulir rilis versi terbaru dan menyimpannya ke dalam basis data, yang diakhiri dengan umpan balik berupa notifikasi keberhasilan proses.



Gambar 3. 7 Activity Diagram Sistem Aplikasi TomatoCare AI



Gambar 3. 8 Diagram Activity Pada Portal Pengguna



Gambar 3. 9 Diagram Activity Pada Dashboard Admin

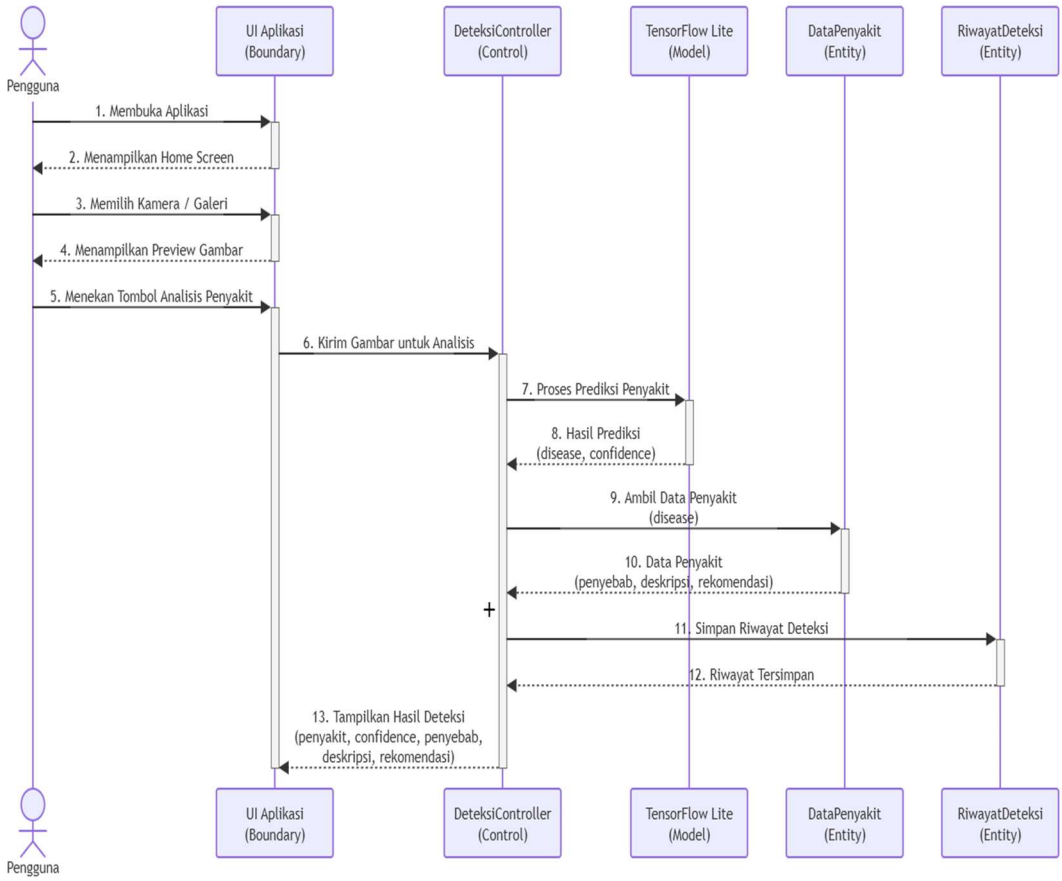
3.4.2.3 *Sequence Diagram*

Sequence Diagram menggambarkan urutan interaksi dan aliran pesan antarobjek atau komponen di dalam sistem berdasarkan urutan waktu eksekusi. Pada perancangan ini, diagram urutan pesan dibagi menjadi dua ruang lingkup, yaitu untuk Aplikasi *Mobile* dan Portal Web Distribusi.

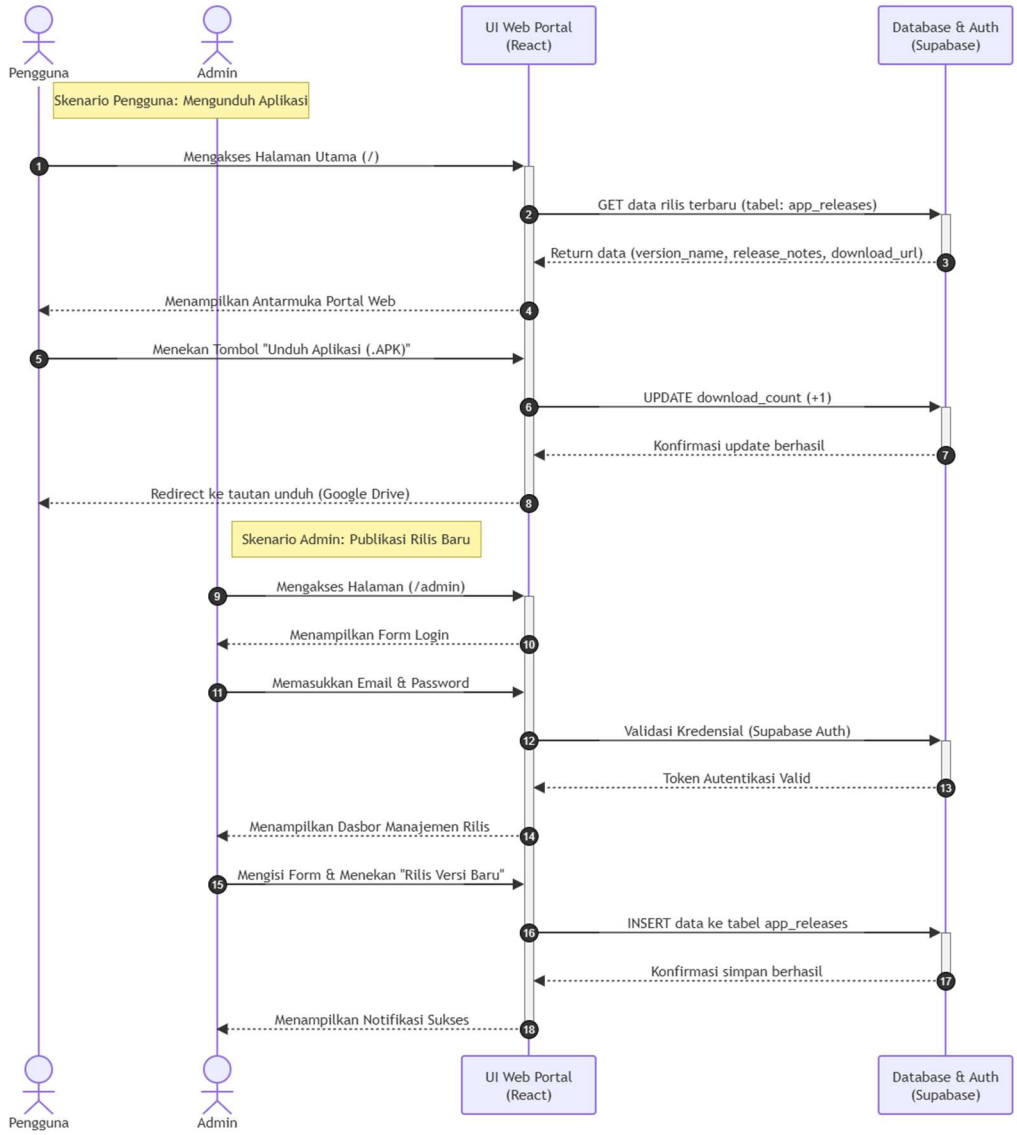
Pada lingkungan Aplikasi *Mobile*, diagram memvisualisasikan interaksi pengguna dalam proses deteksi penyakit tanaman tomat. Aliran pesan dimulai ketika pengguna membuka aplikasi dan memilih sumber gambar melalui kamera atau galeri. Setelah gambar dipilih, aplikasi menampilkan halaman pratinjau

(Preview Screen) agar pengguna dapat memastikan kesesuaian gambar. Saat pengguna menekan tombol analisis, antarmuka akan mengirimkan pesan ke controller, yang kemudian memanggil TFLiteService untuk mengeksekusi proses inferensi menggunakan model *MobileNetV3* secara luring. Hasil prediksi berupa indeks kelas dan tingkat confidence kemudian dicocokkan dengan entitas data penyakit lokal sebelum dikirim ke halaman Result Screen untuk ditampilkan kepada pengguna.

Pada lingkungan Portal Web Distribusi, Sequence Diagram memvisualisasikan komunikasi asinkron antara antarmuka pengguna (*React*) dengan Backend-as-a-Service (*Supabase*). Interaksi ini dibagi menjadi dua skenario utama. Pada skenario Pengguna (petani), saat halaman utama diakses, antarmuka web akan meminta (*fetch*) data rilis terbaru dari basis data. Interaksi lanjutan berupa penekanan tombol unduh akan memicu sistem untuk mengirimkan instruksi pembaruan nilai (*update*) pada kolom download count di basis data, yang kemudian dilanjutkan dengan mengalihkan (*redirect)* pengguna ke tautan penyimpanan berkas eksternal. Pada skenario Admin, urutan interaksi diawali dengan pengiriman data kredensial login untuk divalidasi oleh sistem *Supabase* Auth. Setelah otorisasi dinyatakan valid, Admin dapat mengirimkan pembaruan data rilis aplikasi ke dalam tabel basis data (*insert*), yang diakhiri dengan pengiriman pesan konfirmasi keberhasilan dari sistem kembali ke layar Admin.



Gambar 3. 10 Diagram Sequence pada APK *TomatoCare AI*



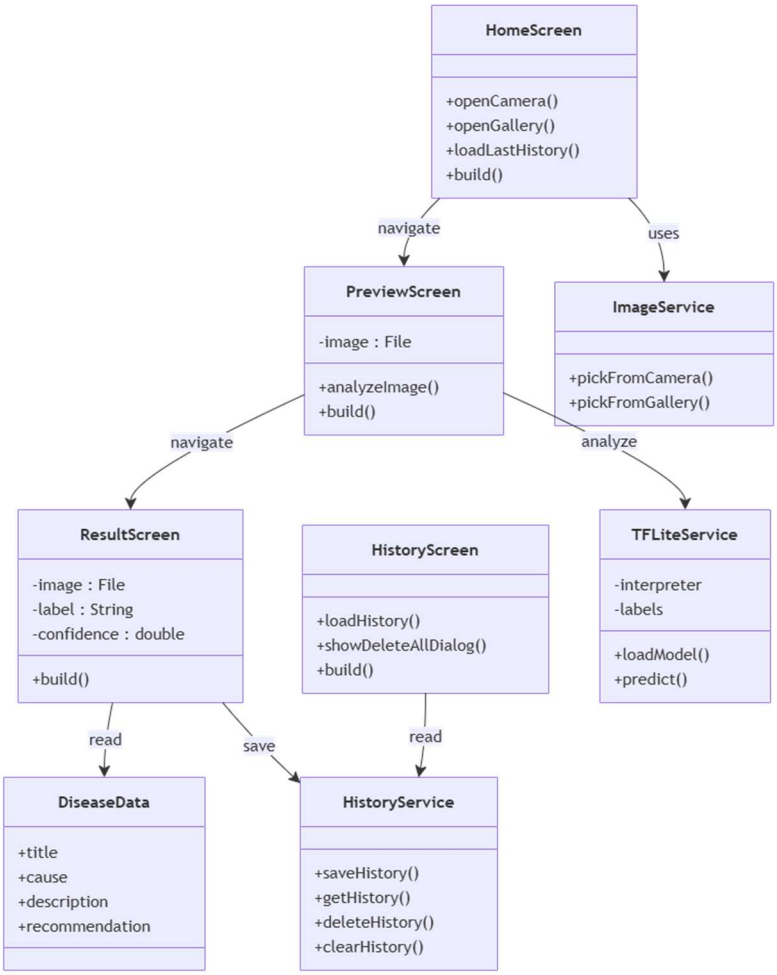
Gambar 3. 11 Diagram Sequence pada Tampilan Dashboard Admin

3.4.2.4 *Class Diagram*

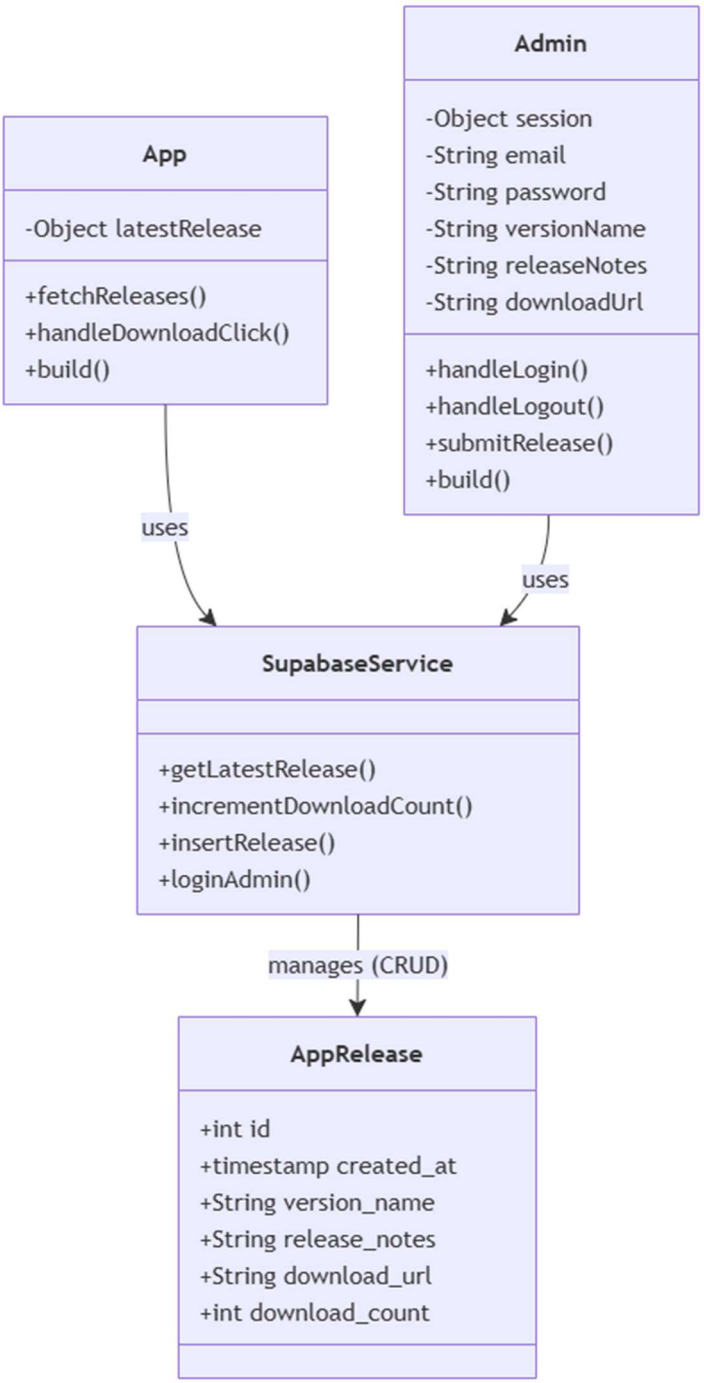
Class Diagram digunakan untuk menggambarkan struktur kelas beserta hubungan antarkelas yang membangun sistem secara keseluruhan. Diagram ini memberikan gambaran mengenai pembagian tugas setiap kelas sehingga proses pengembangan perangkat lunak menjadi lebih terstruktur, modular, dan terorganisasi dengan baik. Pada perancangan sistem ini, struktur kelas dibagi menjadi dua arsitektur, yaitu untuk Aplikasi *Mobile* dan Portal Web Distribusi.

Pada arsitektur Aplikasi *Mobile*, kelas-kelas dirancang untuk mendukung operasional deteksi penyakit secara luring. HomeScreen berfungsi sebagai antarmuka utama yang menyediakan akses pengambilan gambar melalui kamera maupun galeri dengan mendelegasikan tugas pemuatan citra kepada ImageService. Setelah gambar dipilih, pengguna diarahkan ke PreviewScreen untuk mengeksekusi proses analisis menggunakan TFLiteService, yakni kelas yang bertanggung jawab menjalankan inferensi model *MobileNetV3*. Hasil prediksi kemudian diteruskan ke ResultScreen, yang secara otomatis mengambil detail informasi penyakit melalui entitas DiseaseData untuk menyajikan penyebab, deskripsi, dan rekomendasi penanganan. Selain itu, hasil deteksi juga diinstruksikan untuk disimpan secara lokal melalui HistoryService sehingga dapat ditampilkan kembali pada halaman riwayat pengguna

Pada arsitektur Portal Web Distribusi, struktur kelas dirancang dengan memisahkan komponen antarmuka pengguna (UI Component), layanan integrasi (Service), dan entitas data (Entity). Komponen antarmuka direpresentasikan oleh kelas App yang menyajikan portal publik untuk Pengguna, serta kelas Admin yang memuat logika pemrosesan formulir rilis dan autentikasi untuk pengelola. Kedua kelas komponen antarmuka tersebut mendelegasikan proses komunikasi backend kepada kelas *Supabase*Service. Layanan ini secara spesifik bertugas mengeksekusi operasi manipulasi data (CRUD) terhadap kelas entitas AppRelease, yang merupakan representasi pemetaan dari skema tabel app_releases pada infrastruktur basis data *Supabase*..



Gambar 3. 12 *Class Diagram TomatoCare AI*



Gambar 3. 13 Diagram Class Pada Dashboard Admin

3.4.2.5 Skema Basis Data (Entity Relationship Diagram)

Entity Relationship Diagram (ERD) merupakan model konseptual yang digunakan untuk mendeskripsikan struktur logis dari sebuah basis data beserta entitas dan atribut yang menyusunnya. Pada sistem *TomatoCare AI*, pengelolaan data riwayat deteksi di lingkungan aplikasi *mobile* sepenuhnya memanfaatkan penyimpanan lokal (*SharedPreferences*) sehingga tidak memerlukan perancangan basis data relasional. Oleh karena itu, perancangan skema basis data secara khusus diimplementasikan pada lingkungan Portal Web Distribusi.

Pengelolaan data pada portal web distribusi diakomodasi sepenuhnya oleh layanan Backend-as-a-Service (BaaS) *Supabase*. Mengingat fungsinya yang terfokus pada distribusi aplikasi, skema basis data dirancang secara sederhana dan efisien menggunakan satu entitas tabel utama yang dinamakan app_releases. Tabel ini bertugas secara spesifik untuk merekam dan menyimpan seluruh riwayat rilis aplikasi yang dipublikasikan oleh Admin.

Entitas app_releases terdiri dari beberapa atribut utama, yaitu id sebagai kunci primer (*Primary Key*) yang bersifat unik, dan created_at untuk mencatat penanda waktu publikasi (timestamp). Informasi detail terkait aplikasi direpresentasikan oleh atribut version_name untuk identitas versi, release_notes yang memuat teks catatan pembaruan fitur, serta download_url yang menyimpan tautan direktori unduhan berkas instalasi (Google Drive). Selain itu, terdapat atribut download_count bertipe integer yang difungsikan sebagai metrik analitik; atribut ini dirancang agar dapat memicu pembaruan nilai (increment) secara otomatis setiap kali sistem mendeteksi adanya aktivitas pengunduhan oleh Pengguna.

app_releases			
int8	id	PK	Primary Key
timestamptz	created_at		Timestamp
varchar	version_name		Versi Aplikasi
text	release_notes		Catatan Pembaruan
varchar	download_url		Link Unduh (G-Drive)
int4	download_count		Jumlah Unduhan

Gambar 3. 14 Skema Basis Data (Entity Relationship Diagram)

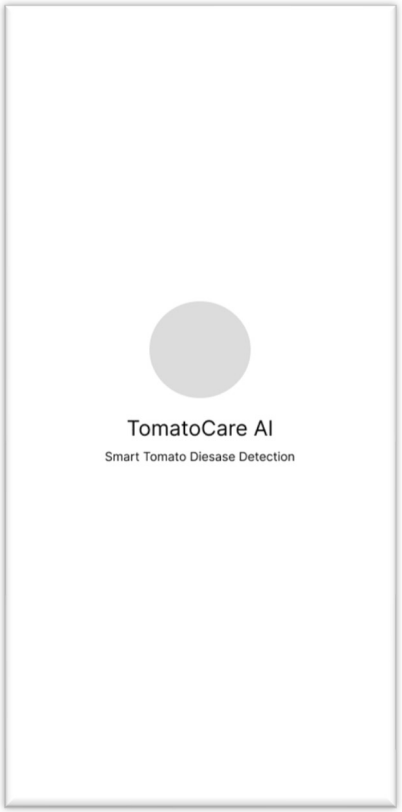
Sumber: Supabase (2026).

3.4.3 Perancangan Antarmuka (*User Interface*)

Perancangan antarmuka (wireframe) dilakukan sebagai tahap awal dalam merancang tampilan aplikasi *TomatoCare AI* sebelum proses implementasi menggunakan *Flutter*. Wireframe berfungsi sebagai gambaran mengenai struktur tata letak (layout), posisi komponen, serta alur navigasi pada setiap halaman aplikasi. Melalui perancangan ini, setiap fitur dapat direncanakan dengan lebih terstruktur sehingga memudahkan proses pengembangan dan implementasi sistem.

Wireframe yang dirancang pada penelitian ini terdiri dari beberapa halaman utama, yaitu *Splash Screen*, *Home Screen*, *Preview Screen*, *Result Screen*, *History Screen*, dan *About Screen*. Masing-masing halaman dirancang sesuai dengan kebutuhan pengguna, mulai dari proses pemilihan gambar, deteksi penyakit tanaman tomat, hingga penyajian hasil deteksi beserta riwayat analisis. Hasil perancangan wireframe selanjutnya diimplementasikan ke dalam aplikasi menggunakan framework *Flutter*.

3.4.3.1 Wireframe *Splash Screen* dan *Home Screen*



Gambar 3. 15 Wireframe *Splash Screen* Gambar 3. 16 Wireframe *Home Screen*

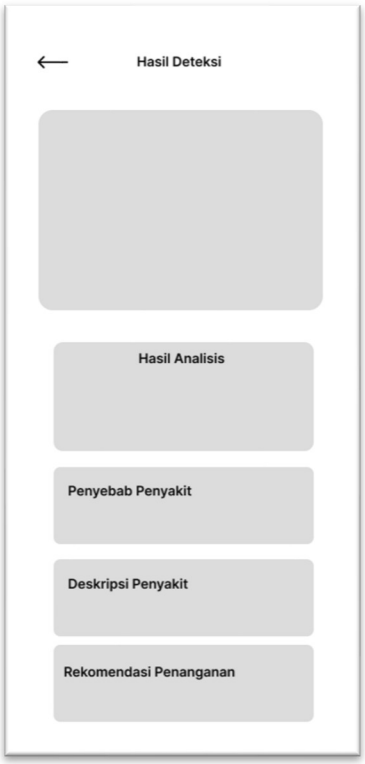
Wireframe *Splash Screen* merupakan tampilan pertama yang muncul saat aplikasi *TomatoCare AI* dijalankan. Halaman ini dirancang sebagai pembuka aplikasi dengan menampilkan identitas berupa logo dan nama aplikasi sebelum pengguna diarahkan ke halaman utama. Selain memberikan identitas aplikasi, halaman ini juga digunakan sebagai proses awal untuk mempersiapkan sistem sebelum seluruh fitur dapat digunakan.

Selanjutnya, *Home Screen* merupakan halaman utama yang digunakan pengguna untuk mengakses seluruh fitur pada aplikasi. Pada halaman ini tersedia tombol untuk memulai proses deteksi penyakit melalui kamera maupun galeri, serta menu menuju halaman riwayat deteksi dan informasi aplikasi. Tata letak halaman

dirancang sederhana agar pengguna dapat memahami fungsi setiap menu dengan mudah tanpa memerlukan petunjuk tambahan.

3.4.3.2 Wireframe *Preview Screen* dan *Result Screen*

Gambar 3. 17 Wireframe *Preview Screen*



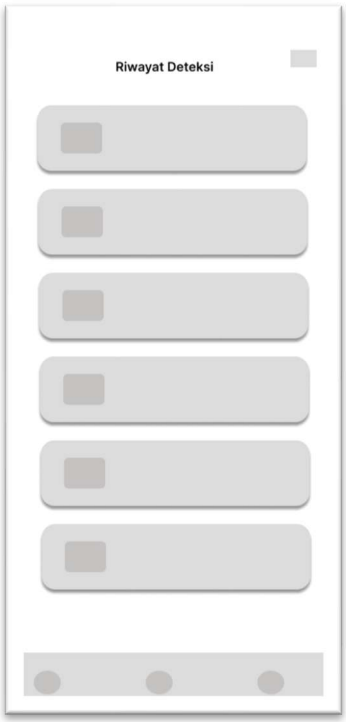
Gambar 3. 18 Wireframe *Result Screen*

Wireframe *Preview Screen* berfungsi untuk menampilkan gambar daun tomat yang telah dipilih oleh pengguna sebelum dilakukan proses deteksi. Halaman ini memberikan kesempatan kepada pengguna untuk memastikan bahwa gambar yang dipilih sudah sesuai sehingga proses klasifikasi dapat dilakukan dengan lebih optimal. Selain itu, tersedia tombol untuk memulai proses analisis menggunakan model *MobileNetV3*.

Setelah proses deteksi selesai dilakukan, aplikasi akan menampilkan *Result Screen*. Pada halaman ini pengguna dapat melihat hasil klasifikasi berupa nama

penyakit, nilai confidence, serta rekomendasi penanganan berdasarkan hasil prediksi model. Seluruh informasi disusun dalam satu tampilan agar pengguna dapat memahami kondisi tanaman dengan cepat dan mudah.

3.4.3.3 Wireframe *History Screen* dan *About Screen*



Gambar 3. 19 Wireframe *History Screen*

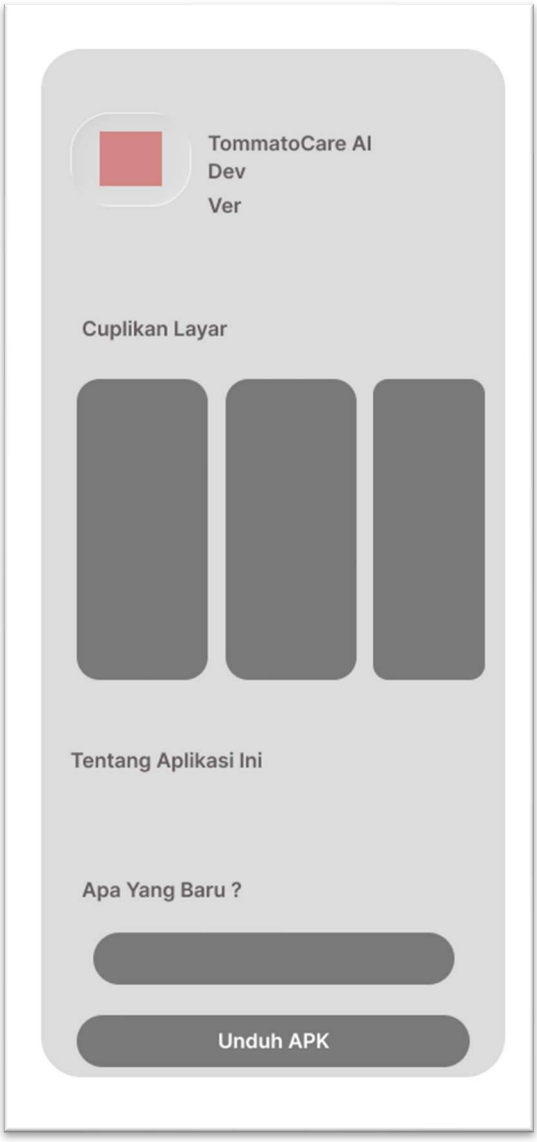
Gambar 3. 20 Wireframe *About Screen*

Wireframe *History Screen* digunakan untuk menampilkan riwayat hasil deteksi yang telah dilakukan sebelumnya. Setiap riwayat memuat informasi mengenai jenis penyakit, nilai confidence, serta waktu pelaksanaan deteksi. Halaman ini juga menyediakan fitur untuk menghapus riwayat, baik secara satu per satu maupun seluruhnya, sehingga pengguna dapat mengelola data hasil deteksi sesuai kebutuhan.

Sementara itu, *About Screen* berisi informasi mengenai aplikasi *TomatoCare AI*, seperti deskripsi singkat aplikasi, tujuan pengembangan, teknologi

yang digunakan, serta informasi mengenai pengembang. Halaman ini bertujuan untuk memberikan informasi tambahan kepada pengguna mengenai aplikasi yang digunakan sekaligus menjelaskan bahwa sistem dikembangkan sebagai media deteksi penyakit tanaman tomat berbasis *Deep Learning MobileNetV3*.

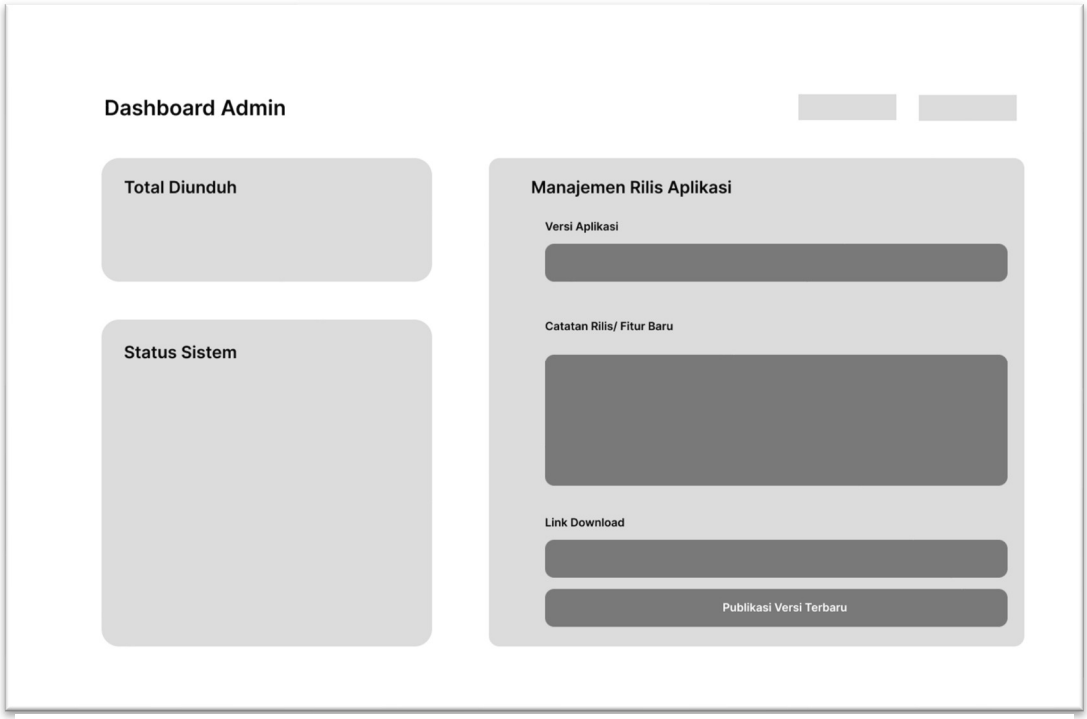
3.4.3.4 Wireframe Portal Web Distribusi



Gambar 3. 21 Wireframe Halaman Utama Portal Web (Publik)

Wireframe Halaman Utama Portal Web digunakan untuk menampilkan portal distribusi dan rilis terbaru dari aplikasi *TomatoCare AI*. Halaman ini memuat informasi mengenai identitas aplikasi, cuplikan layar antarmuka (screenshot), serta catatan pembaruan versi (release notes). Halaman ini juga menyediakan fitur tombol aksi utama untuk mengunduh APK, sehingga pengguna dapat memperoleh berkas instalasi dan memasang aplikasi versi terbaru ke dalam perangkat seluler mereka dengan mudah.

3.4.3.5 *Wireframe* Halaman Dashboard Admin



Gambar 3. 22 Wireframe Halaman Dashboard Admin

Wireframe Halaman Dasbor Admin dirancang khusus sebagai pusat kendali bagi administrator untuk memantau dan mengelola data distribusi aplikasi *TomatoCare AI*. Pada antarmuka ini, terdapat panel pemantauan di sisi kiri yang menampilkan informasi analitik berupa akumulasi total aplikasi diunduh, serta indikator status sistem untuk memantau konektivitas basis data. Sementara itu, pada sisi utama disediakan area formulir Manajemen Rilis Aplikasi yang memungkinkan pengelola untuk melakukan pembaruan data, meliputi input versi aplikasi, catatan rilis atau fitur baru, serta tautan unduhan (link download). Halaman ini juga

dilengkapi dengan tombol aksi untuk memublikasikan versi terbaru ke sistem, serta navigasi di bagian atas untuk meninjau portal publik atau keluar dari sesi (logout), sehingga proses distribusi dan pembaruan aplikasi dapat dikelola secara terpusat dan efisien.