Jurnal Ilmiah MEDIA SISFO
Vol. 19, No. 2, Oktober 2025

ISSN: 1978-8126
e-ISSN: 2527-7340

Implementasi Aplikasi Android Deteksi Penyakit Tanaman Tomat Berbasis *Cloud* menggunakan MVVM

M. Edoazani[1], Syahril Rizal[2], Nurul Adha Octarini Saputri[3], M. Soekarno Putra[4]*

Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Bina Darma
Jl. Jenderal Ahmad Yani No.3, 9/10 Ulu, Kecamatan Seberang Ulu I, Kota Palembang, Sumatera Selatan, Indonesia
m.edoazani@gmail.com [1], syahril.rizal@binadarma.ac.id [2], nuruladhaos@binadarma.ac.id [3], soekarno@binadarma.ac.id [4]

Submitted : **13/09/2025**; Reviewed : **17/09/2025**; Accepted : **30/10/2025**; Published : **31/10/2025**

Abstract

Tomato plants as important horticultural commodities in Indonesia are vulnerable to various diseases that are difficult to distinguish visually, causing treatment delays and economic losses for farmers. This research develops an Android application for cloud-based tomato plant disease detection using Model-View-ViewModel (MVVM) architecture with Jetpack Compose framework. Extreme Programming (XP) methodology is applied through four main stages: planning, design, coding, and testing iteratively to ensure responsive development toward system requirements. The system integrates MobileNetV2 machine learning model deployed on Azure App Service with Firebase Authentication and Firestore for authentication and detection history storage. Dataset containing 16,012 tomato leaf images with 10 disease classes is used to train the model achieving 91.74% accuracy. Cloud-hybrid architecture enables detection processes to be performed on server without burdening mobile devices with average response time of 3 seconds. Black Box Testing shows all application features function according to specifications with 100% success rate. Research results prove that cloud computing integration with mobile application development can provide practical and scalable solutions to help farmers perform tomato plant disease detection quickly and accurately.

Keywords : android, cloud computing, extreme programming, machine learning, mvvm, plant disease detection

Abstrak

Tanaman tomat sebagai komoditas hortikultura penting di Indonesia rentan terhadap berbagai penyakit yang sulit dibedakan secara visual, menyebabkan keterlambatan penanganan dan kerugian ekonomi bagi petani. Penelitian ini mengembangkan aplikasi Android untuk deteksi penyakit tanaman tomat berbasis *cloud* menggunakan arsitektur Model-View-ViewModel (MVVM) dengan *framework Jetpack Compose*. Metodologi *Extreme Programming* (XP) diterapkan melalui empat tahap utama *planning*, *design*, *coding*, dan *testing* secara iteratif untuk memastikan pengembangan yang responsif terhadap kebutuhan sistem. Sistem mengintegrasikan model *machine learning MobileNetV2* yang di-*deploy* di *Azure App Service* dengan *Firebase Authentication* dan *Firestore* untuk autentikasi serta penyimpanan riwayat deteksi. Dataset berisi 16.012 citra daun tomat dengan 10 kelas penyakit digunakan untuk melatih model yang mencapai akurasi 91,74%. Arsitektur *cloud-hybrid* memungkinkan proses deteksi dilakukan di server tanpa membebani perangkat *mobile* dengan response time rata-rata 3 detik. Pengujian *Black Box Testing* menunjukkan seluruh fitur aplikasi berfungsi sesuai spesifikasi dengan tingkat keberhasilan 100%. Hasil penelitian membuktikan bahwa integrasi *cloud computing* dengan pengembangan aplikasi *mobile* dapat memberikan solusi praktis dan *scalable* untuk membantu petani melakukan deteksi penyakit tanaman tomat secara cepat dan akurat.

Kata kunci : android, *cloud computing*, deteksi penyakit tanaman, *extreme programming*, *machine learning*, mvvm

1. Pendahuluan

Perkembangan teknologi telah membawa dampak signifikan di berbagai sektor, termasuk pertanian. Inovasi di bidang kecerdasan buatan (*Artificial Intelligence*), pemrosesan citra digital, dan teknologi mobile telah memungkinkan berbagai solusi cerdas untuk meningkatkan efisiensi dan produktivitas pertanian [1]. Dalam konteks pertanian modern, pemanfaatan teknologi informasi diharapkan dapat membantu petani dalam mengidentifikasi permasalahan tanaman secara lebih cepat dan akurat, sehingga mampu mendukung proses pengambilan keputusan yang lebih baik di lapangan [2].

Tanaman tomat (*Solanum lycopersicum*) merupakan salah satu komoditas hortikultura penting yang memiliki nilai ekonomi tinggi dan banyak dibudidayakan di Indonesia [3]. Data dari Badan Pusat Statistik 2023 menunjukkan bahwa produksi tomat di Indonesia mencapai 1,1 juta ton. Namun, tanaman tomat sangat rentan terhadap serangan berbagai penyakit seperti bercak daun septoria (*septoria leaf spot*), busuk

ISSN: 1978-8126
e-ISSN: 2527-7340

daun (*late blight*), bercak melingkar (*target spot*), dan virus keriting daun kuning (*yellow leaf curl virus*) [4]. Kesulitan petani dalam membedakan jenis penyakit secara visual sering menyebabkan keterlambatan penanganan yang berakibat pada kerugian ekonomi akibat gagal panen [5].

Beberapa penelitian sebelumnya telah mengembangkan solusi deteksi penyakit tanaman melalui aplikasi berbasis Android. Anwari [6] mengembangkan aplikasi klasifikasi penyakit daun tomat menggunakan arsitektur *SqueezeNet* dalam model CNN dengan akurasi 86,9%, namun, seluruh proses klasifikasi masih dijalankan secara lokal (*on-device*). Saputra [7] mengembangkan aplikasi Android untuk deteksi penyakit tanaman padi menggunakan *Teachable Machine*, yang terbukti efektif namun belum mengintegrasikan layanan *cloud computing*. Keterbatasan pendekatan *on-device processing* terletak pada performa aplikasi yang sangat bergantung pada spesifikasi perangkat yang digunakan, tidak scalable untuk *deployment* massal, dan *maintenance* yang kompleks.

Pengembangan aplikasi berbasis *mobile*, khususnya di platform Android yang memiliki penetrasi luas di masyarakat, menjadi solusi ideal untuk menjembatani kesenjangan antara teknologi canggih dan kebutuhan praktis petani di lapangan [8]. Pemanfaatan layanan *cloud computing* memungkinkan proses analisis citra dilakukan secara cepat dan akurat tanpa membebani perangkat pengguna [9]. Model *machine learning* yang di-*deploy* di *cloud* dapat menangani proses klasifikasi secara efisien, sementara aplikasi Android berperan sebagai antarmuka pengguna yang intuitif dan mudah digunakan [10].

Penelitian ini mengembangkan aplikasi Android untuk deteksi penyakit tanaman tomat berbasis *cloud* dengan implementasi arsitektur Model-View-ViewModel (MVVM) menggunakan *framework Jetpack Compose*. Metodologi Extreme Programming (XP) diterapkan melalui empat tahap utama: Planning, Design, Coding, dan Testing, untuk memastikan pengembangan yang iteratif dan responsif terhadap kebutuhan pengguna. Sistem terintegrasi dengan FastAPI *backend*, *Firebase Authentication*, dan model MobileNetV2 yang di-*host* di *Azure* untuk memberikan solusi deteksi penyakit yang komprehensif dan *scalable*.

Fokus utama penelitian adalah pada aspek rekayasa perangkat lunak, yaitu proses integrasi model machine learning ke dalam platform Android, optimasi kinerja aplikasi agar berjalan dengan baik di perangkat mobile, serta perancangan antarmuka pengguna yang intuitif dan mudah digunakan oleh petani. Dengan demikian, tujuan dari penelitian ini adalah untuk mengembangkan aplikasi android yang mampu mengidentifikasi penyakit pada tanaman tomat secara otomatis dengan memanfaatkan layanan cloud computing, guna membantu petani dalam pengambilan keputusan dan pengendalian penyakit secara lebih cepat dan akurat.

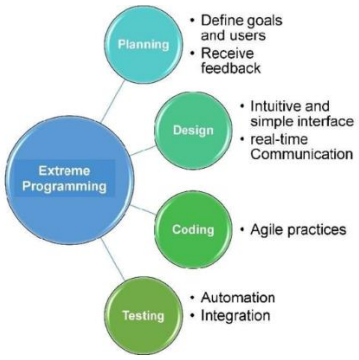
2. Metodologi

Penelitian ini menggunakan metodologi penelitian dan pengembangan (*Research and Development*) dengan pendekatan *Extreme Programming* (XP). Metodologi XP dipilih karena menekankan pada pengembangan iteratif, kolaborasi intensif antara pengembang dan pengguna, serta pengujian berkelanjutan yang sangat sesuai untuk pengembangan aplikasi *mobile* yang dinamis dan membutuhkan adaptasi cepat terhadap perubahan kebutuhan [11].

2.1. Tahapan Pengembangan Extreme Programming

Siklus pengembangan XP terdiri dari empat tahap utama yang dilakukan secara berulang hingga aplikasi memenuhi kebutuhan pengguna dan mencapai tingkat kualitas yang diinginkan.

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 1. *Metodologi Extreme Programming*

2.1.1. Perencanaan (Planning)

Tahap planning dilakukan melalui diskusi dengan dosen pembimbing serta kajian pustaka yang relevan, tanpa melakukan wawancara atau keterlibatan langsung dengan pengguna akhir. Identifikasi kebutuhan dan fitur utama aplikasi didasarkan pada analisis permasalahan dari literatur yang telah dikaji dan arahan dari dosen pembimbing. Fokus perencanaan adalah merumuskan fitur yang mendukung tujuan penelitian, meliputi kemampuan unggah foto daun tomat untuk dianalisis, deteksi penyakit menggunakan model CNN yang dikirim ke API *cloud*, menampilkan hasil deteksi dengan informasi lengkap penyakit, penyimpanan riwayat deteksi di *Firebase Firestore*, akses berita pertanian untuk informasi terkini, dan pengaturan akun pengguna.

2.1.2. Perancangan (Design)

Pada tahap desain, dilakukan dengan pendekatan iteratif untuk memastikan semua fitur yang direncanakan dapat diimplementasikan dengan baik. Desain sistem mencakup arsitektur aplikasi menggunakan pola Model-View-ViewModel (MVVM) dengan *Jetpack Compose* untuk antarmuka pengguna yang responsif. Selain itu, dirancang pula API berbasis FastAPI untuk menangani komunikasi dengan model *machine learning* di *cloud*, integrasi *Firebase Authentication* untuk autentikasi pengguna dan *Firestore* untuk penyimpanan riwayat deteksi, serta perancangan UML diagram yang mencakup *Use Case*, *Activity*, dan *Sequence* Diagram untuk memvisualisasikan struktur dan alur kerja aplikasi.

2.1.3. Pengkodean (Coding)

Tahap *coding* merupakan proses penerjemahan desain sistem menjadi kode program yang dapat dijalankan. Implementasi dilakukan secara iteratif sesuai prinsip XP dengan pengembangan aplikasi Android berbasis Kotlin menggunakan *Jetpack Compose* sebagai *UI Toolkit modern*. Penerapan arsitektur MVVM dengan *separation of concerns* dilakukan untuk maintainability yang lebih baik. Integrasi *Firebase* SDK untuk *Authentication* dan *Firestore*, komunikasi dengan *backend* FastAPI menggunakan Retrofit untuk konsumsi REST API, serta pemanggilan model CNN MobileNetV2 yang di-host di Azure untuk melakukan prediksi penyakit daun tomat.

2.1.4. Pengujian (Testing)

Pengujian merupakan tahapan penting untuk memastikan sistem berjalan sesuai kebutuhan fungsional. Metode pengujian yang digunakan adalah *Black Box Testing*, yang berfokus pada masukan dan keluaran sistem tanpa memperhatikan struktur internal kode program. Ruang lingkup pengujian meliputi autentikasi pengguna melalui *Firebase Authentication*, fitur deteksi penyakit tomat pada *endpoint* predict di *backend*, penyimpanan dan penampilan riwayat deteksi pada *Firestore*, fitur berita pertanian, dan fitur pengaturan serta *logout*.

ISSN: 1978-8126
e-ISSN: 2527-7340

2.2. Perancangan Sistem

2.2.1. Diagram UML

Perancangan sistem menggunakan diagram UML untuk memvisualisasikan struktur dan alur kerja aplikasi. *Use Case Diagram* menggambarkan fungsionalitas sistem dan interaksi antara pengguna dengan sistem aplikasi, mencakup skenario autentikasi, deteksi penyakit, manajemen riwayat, dan akses berita pertanian. *Activity Diagram* menggambarkan alur aktivitas atau proses bisnis dalam sistem, khususnya *sequence of activities* dan *decision points* dalam proses deteksi penyakit mulai dari pengambilan gambar hingga penyimpanan hasil. *Sequence Diagram* menggambarkan interaksi antar objek berdasarkan urutan waktu, menunjukkan message passing antar komponen sistem dalam proses autentikasi pengguna dan komunikasi aplikasi-API-database.

2.2.2. Arsitektur Sistem

Sistem dikembangkan dengan arsitektur *cloud-hybrid* yang mengombinasikan layanan *Firebase* dan Microsoft Azure. Aplikasi Android menggunakan arsitektur MVVM dengan struktur layer yang terorganisir meliputi presentation layer yang dibangun dengan *Jetpack Compose* untuk UI yang responsif, domain layer yang memuat model bisnis logika, data layer yang mengatur akses data dari API dan *Firebase* menggunakan *Repository pattern*, serta *dependency injection* menggunakan Hilt untuk mengelola dependensi antar komponen. *Backend* API dibangun menggunakan FastAPI dengan Python yang berperan sebagai penghubung antara aplikasi Android dengan model pembelajaran mesin, menyediakan *endpoint* predict untuk prediksi penyakit, API berita, autentikasi dan manajemen riwayat pengguna, serta verifikasi *Firebase* ID Token untuk keamanan akses.

2.3. Model Machine Learning

Penelitian ini menggunakan model deteksi penyakit daun tomat berbasis MobileNetV2, sebuah arsitektur *Convolutional Neural Network* (CNN) yang ringan namun berperforma tinggi. MobileNetV2 dipilih karena memiliki jumlah parameter yang relatif sedikit sehingga cocok untuk *deployment* di *cloud* dengan *response time* yang cepat, mendukung ukuran input citra standar 224×224 piksel, dan telah terbukti efektif dalam berbagai penelitian deteksi objek dan klasifikasi citra tanaman. Dataset yang digunakan bersumber dari penelitian terdahulu berisi 16.012 citra daun tomat yang terbagi menjadi 10 kelas penyakit dengan *preprocessing* data meliputi perubahan resolusi ke 224×224 piksel, normalisasi nilai piksel dari 0-255 menjadi 0-1, serta augmentasi data pada *training* set untuk mencegah *overfitting*.

3. Hasil dan Pembahasan

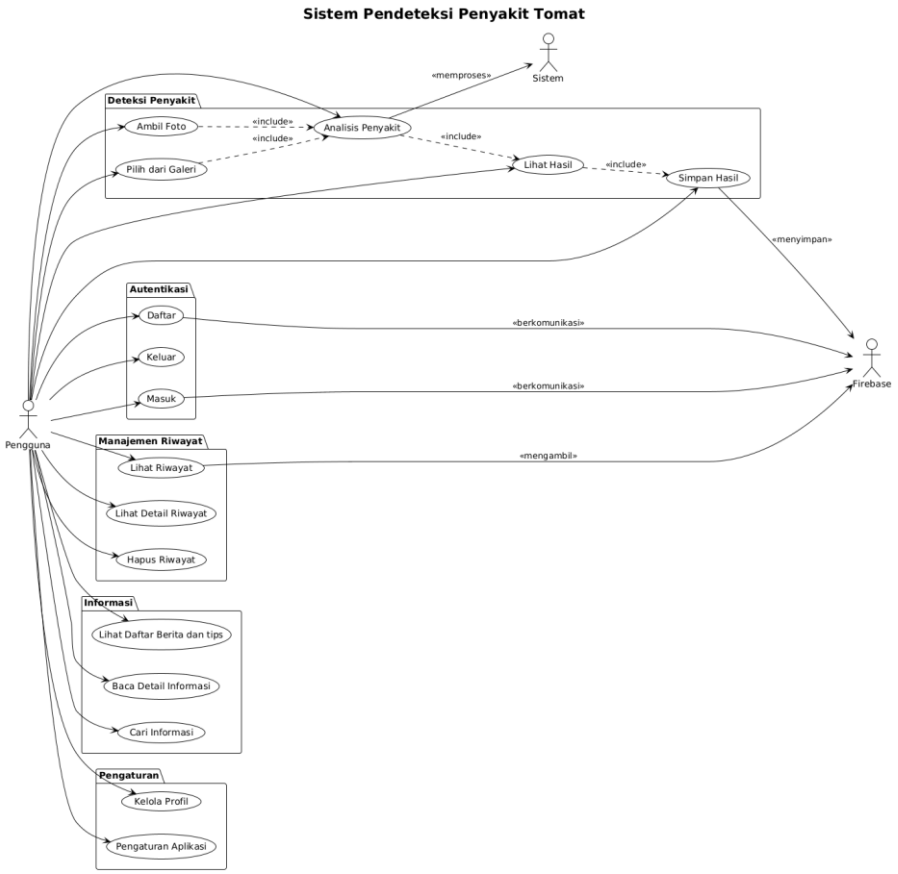
3.1. Perancangan Sistem

Perancangan sistem menggunakan diagram UML untuk memvisualisasikan struktur dan alur kerja aplikasi deteksi penyakit tanaman tomat berbasis *cloud*. Diagram UML yang digunakan meliputi *Use Case Diagram, Activity Diagram, dan Sequence Diagram* untuk membantu merancang arsitektur sistem yang jelas dan terstruktur.

3.1.1. Use Case Diagram

Use Case Diagram digunakan untuk menggambarkan fungsionalitas sistem dan interaksi antara pengguna (aktor) dengan sistem aplikasi, mencakup skenario autentikasi, deteksi penyakit, manajemen riwayat, dan akses konten pertanian.

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 2. *Use Case Diagram Aplikasi Pendeteksi Penyakit Tomat*

Pada gambar 1 menunjukkan interaksi antara Pengguna sebagai aktor utama dengan berbagai fungsi yang tersedia dalam aplikasi. Sistem memiliki enam kelompok fungsi utama yang dapat diakses oleh pengguna:

a. Autentikasi Pengguna

Pengguna dapat melakukan proses masuk dengan memasukkan email dan password yang telah terdaftar melalui *Firebase Authentication* untuk mengakses sistem aplikasi. Untuk pengguna baru, tersedia fitur daftar yang memungkinkan pembuatan akun dengan mengisi data lengkap seperti nama, email, dan password. Setelah selesai menggunakan aplikasi, pengguna dapat melakukan keluar untuk mengakhiri sesi dan menghapus data sensitif dari perangkat.

b. Deteksi Penyakit

Pengguna dapat mengambil gambar daun tomat langsung menggunakan kamera perangkat dengan izin akses kamera yang diperlukan untuk proses analisis. Alternatif lain, pengguna juga dapat memilih gambar dari galeri perangkat dengan format JPG, JPEG, atau PNG dengan ukuran maksimal 2 MB. Setelah gambar dipilih, sistem melakukan analisis penyakit dengan mengirim gambar ke API cloud yang menggunakan model MobileNetV2 untuk prediksi. Hasil analisis kemudian ditampilkan kepada pengguna berupa diagnosis penyakit, tingkat *confidence*, gejala, penyebab, dan solusi penanganan. Hasil deteksi secara otomatis disimpan ke *Firebase Firestore* untuk riwayat pengguna.

c. Manajemen Riwayat

Pengguna dapat melihat daftar lengkap hasil deteksi yang pernah dilakukan dengan data yang diurutkan berdasarkan tanggal terbaru dari *Firebase Firestore*. Untuk setiap entri riwayat, pengguna dapat melihat informasi detail lengkap termasuk gambar yang digunakan dan hasil diagnosis yang diperoleh. Pengguna juga memiliki opsi untuk menghapus entri riwayat tertentu yang tidak diperlukan lagi dari database.

ISSN: 1978-8126
e-ISSN: 2527-7340

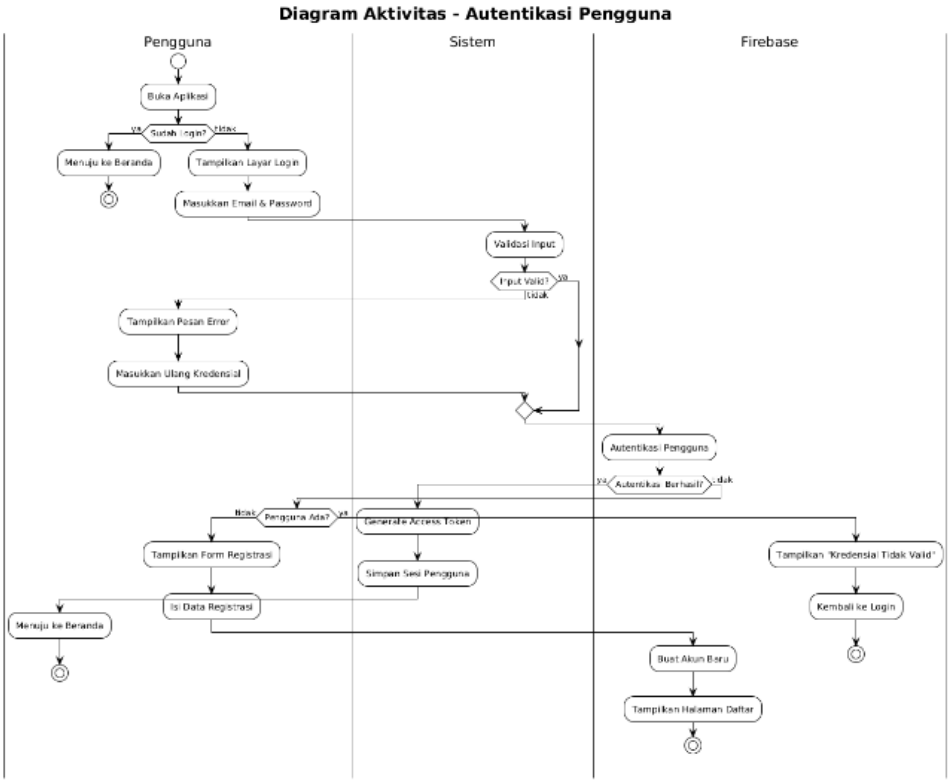
d. Konten Pertanian
Sistem menampilkan daftar berita dan tips pertanian terkini yang diambil dari *backend* API untuk memberikan informasi edukatif kepada pengguna. Pengguna dapat membaca konten lengkap dari berita atau tips yang dipilih untuk mendapatkan wawasan lebih mendalam tentang pertanian tomat. Tersedia juga fitur pencarian yang memungkinkan pengguna mencari konten berdasarkan kata kunci tertentu untuk menemukan informasi yang spesifik.

e. Pengaturan Aplikasi
Pengguna dapat mengubah preferensi aplikasi dan mengelola profil akun sesuai kebutuhan melalui menu pengaturan yang tersedia.

3.1.2. Activity Diagram

Activity Diagram menggambarkan alur aktivitas atau proses bisnis dalam sistem aplikasi deteksi penyakit tanaman tomat. Diagram ini menunjukkan proses utama yaitu autentikasi pengguna, dan deteksi penyakit.

a. Autentikasi Pengguna
Alur dimulai ketika pengguna membuka aplikasi, sistem memeriksa keberadaan sesi login yang valid di penyimpanan lokal perangkat. Jika sesi ditemukan dan masih berlaku, pengguna langsung diarahkan ke beranda aplikasi. Sebaliknya, jika tidak ada sesi valid, sistem menampilkan layar login yang mengharuskan pengguna memasukkan email dan password. Sistem melakukan validasi format input, kemudian mengirimkan permintaan autentikasi ke *Firebase Authentication*. Jika kredensial valid, *Firebase* mengembalikan access token dan data profil yang disimpan sistem untuk sesi berikutnya, lalu mengarahkan pengguna ke beranda. Untuk kasus autentikasi gagal, sistem menampilkan pesan error dan memberikan opsi untuk mencoba lagi atau mendaftar akun baru.

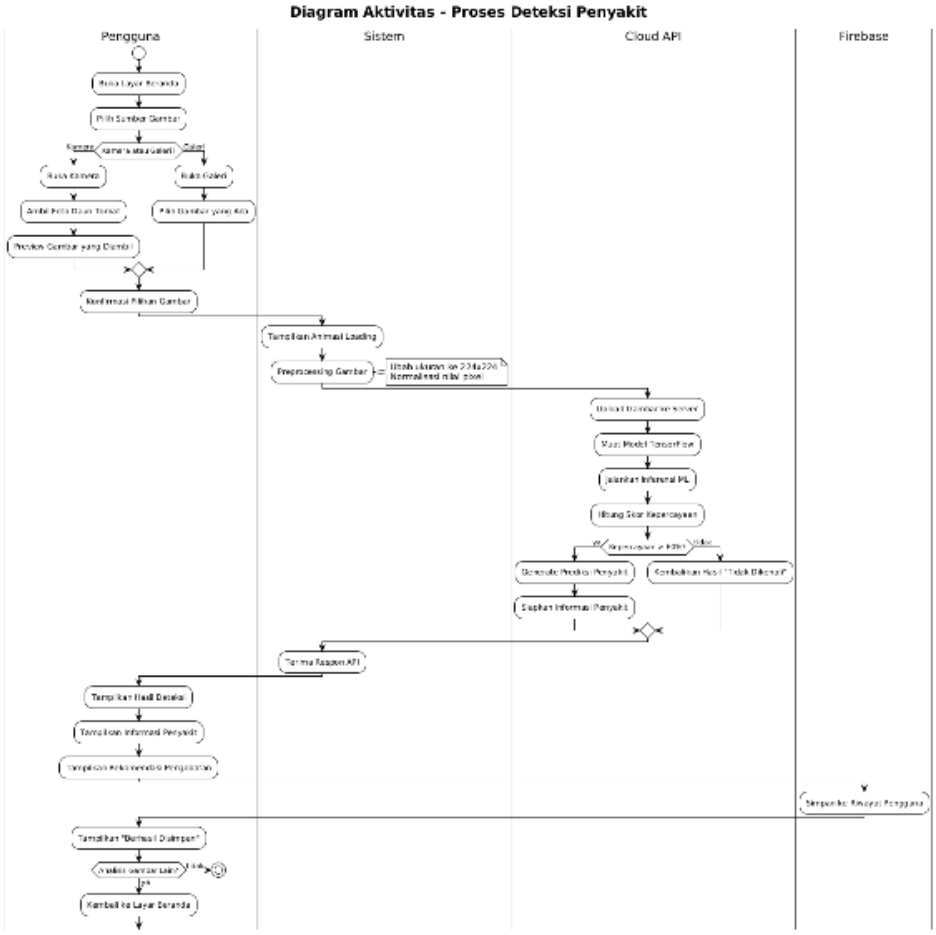


Gambar 3. *Activity Diagram Autentikasi Pengguna*

b. Deteksi Penyakit
Pengguna memilih sumber gambar melalui kamera atau galeri perangkat, sistem membuka interface yang sesuai untuk pengambilan atau pemilihan gambar daun tomat. Setelah gambar dikonfirmasi,

ISSN: 1978-8126
e-ISSN: 2527-7340

sistem menampilkan animasi loading dan melakukan *preprocessing* dengan mengubah ukuran gambar ke 224x224 piksel serta normalisasi nilai pixel. Gambar yang telah diproses diunggah ke server *cloud* melalui *endpoint* predict dengan autentikasi Firebase ID token. Server memvalidasi file, memuat model MobileNetV2, dan menjalankan inferensi untuk menghasilkan prediksi penyakit. Sistem menghitung tingkat kepercayaan dan melakukan pengecekan *threshold* 60%. Jika *confidence score* memenuhi *threshold*, sistem mengembalikan hasil deteksi lengkap dengan informasi penyakit, gejala, penyebab, dan solusi penanganan, kemudian menyimpan hasil ke *Firebase Firestore* sebagai riwayat pengguna.

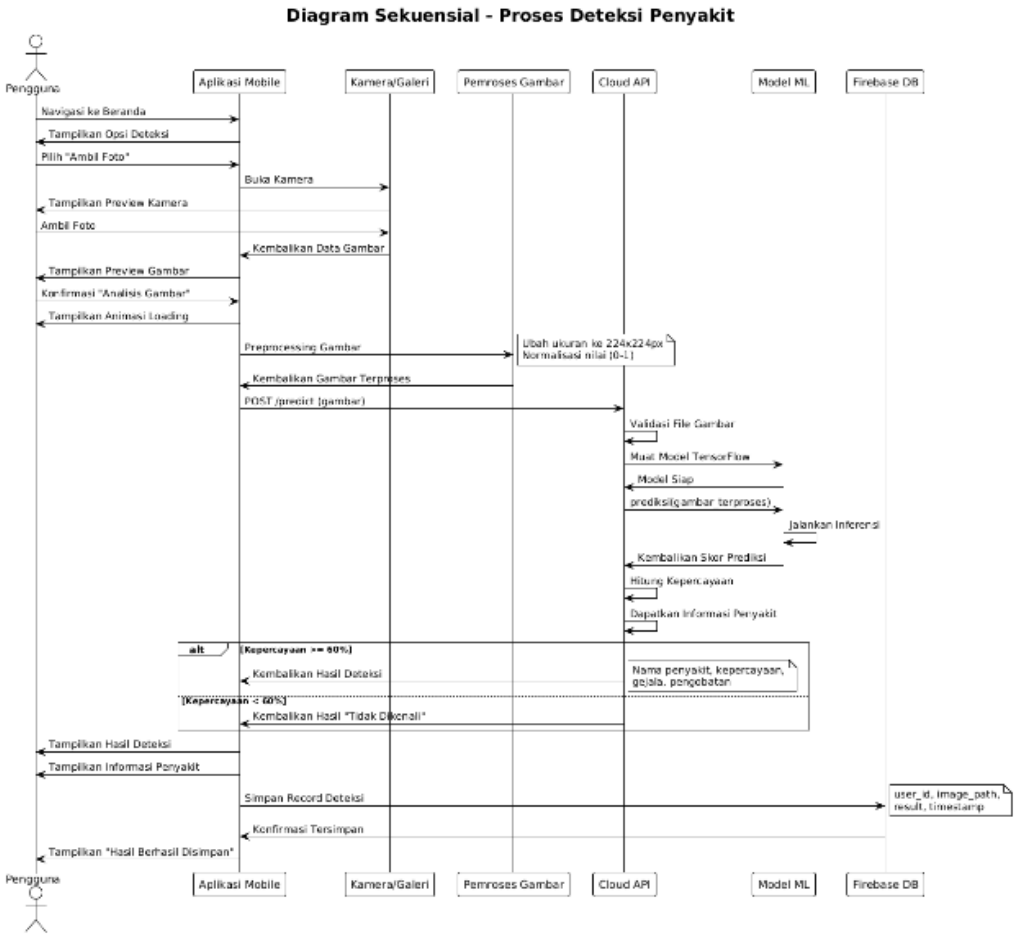


Gambar 4. *Activity Diagram Deteksi Penyakit*

3.1.3. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar objek dalam sistem berdasarkan urutan waktu (*timeline*), menunjukkan message passing antar komponen sistem dalam aplikasi deteksi penyakit tanaman tomat. Diagram ini memvisualisasikan komunikasi yang terjadi antara pengguna, aplikasi *mobile*, layanan *cloud*, dan database dalam tiga skenario utama sistem.

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 5. *Squence Diagram Proses Deteksi Penyakit*

3.2. Hasil Antarmuka Sistem

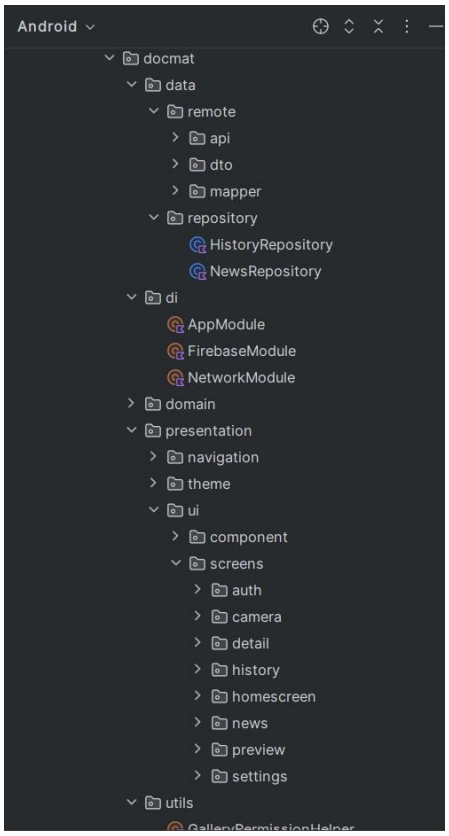
Penelitian ini telah berhasil mengimplementasikan aplikasi Android untuk deteksi penyakit tanaman tomat berbasis cloud menggunakan metodologi *Extreme Programming* (XP). Sistem yang dikembangkan terdiri dari dua komponen utama:

3.2.1. Implementasi Antarmuka Aplikasi Android

Aplikasi dikembangkan menggunakan arsitektur Model-View-ViewModel (MVVM) dengan *Jetpack Compose* untuk memberikan pengalaman pengguna yang responsif dan modern. Struktur implementasi mengikuti prinsip *separation of concerns* dengan pembagian layer sebagai berikut:

1. Struktur Layer Aplikasi
   Struktur layer aplikasi dirancang mengikuti arsitektur Model–View–ViewModel (MVVM) yang memisahkan tanggung jawab antar komponen agar sistem lebih mudah dikembangkan dan dipelihara. Setiap layer memiliki peran spesifik dalam mengelola data, logika bisnis, serta antarmuka pengguna. Pendekatan ini memungkinkan proses integrasi antara Firebase, API FastAPI, dan komponen UI Jetpack Compose berlangsung secara terorganisir, efisien, serta mendukung prinsip *separation of concerns*.

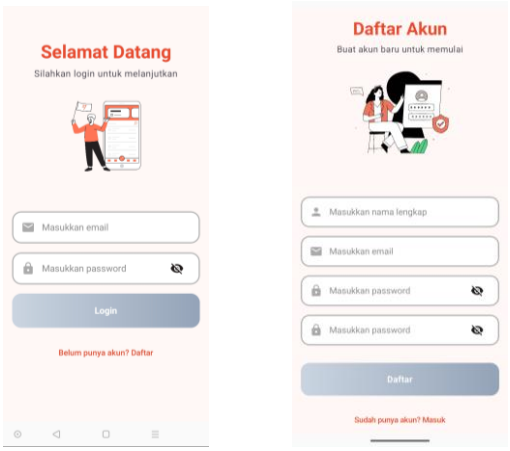
ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 6. *Struktur layer aplikasi*

a. DI (*Dependency Injection*) - konfigurasi Hilt untuk dependensi Retrofit, OkHttp, Firebase, dan Repository
b. Data Layer - mengatur akses data dari API dan Firebase, termasuk service interface dan DTO
c. Domain Layer - model domain sebagai representasi logis data aplikasi
d. Presentation Layer - antarmuka pengguna dengan Jetpack Compose, dikelompokkan berdasarkan screen
e. Utils - fungsi pembantu untuk izin akses dan pemotongan gambar

2. Fitur-fitur Utama yang Diimplementasikan:
   a. Halaman Login dan *Register*
   Halaman ini merupakan pintu masuk utama pengguna untuk mengakses aplikasi. Proses autentikasi dilakukan menggunakan *Firebase Authentication*, yang menjamin keamanan data pengguna serta kemudahan login dengan verifikasi email dan password.



Gambar 7. *halaman login dan register menggunakan firebase authentication*

ISSN: 1978-8126
e-ISSN: 2527-7340

Gambar tersebut menunjukkan antarmuka login dan register yang dibangun menggunakan Jetpack Compose. Pengguna dapat masuk dengan akun yang telah terdaftar atau membuat akun baru melalui form pendaftaran. Proses validasi input dilakukan secara *real-time*, dan sistem menampilkan pesan kesalahan jika format email atau password tidak valid.

b. Halaman Utama

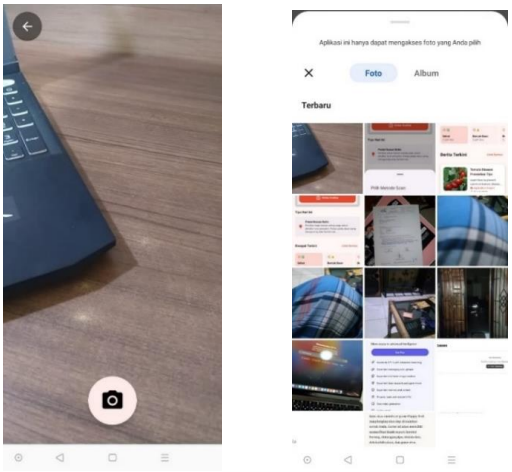
Halaman utama menampilkan menu navigasi ke seluruh fitur aplikasi seperti deteksi penyakit, riwayat, berita pertanian, dan pengaturan akun.

Gambar 8. *Halaman Utama Dashboard*

Tampilan pada gambar memperlihatkan dashboard utama dengan ikon navigasi di bagian bawah layar. Pengguna dapat langsung memilih fitur yang diinginkan dengan satu sentuhan. Desain dibuat sederhana agar ramah bagi pengguna non-teknis, khususnya petani.

c. Halaman memilih sumber gambar

Halaman ini digunakan untuk memilih sumber citra daun tomat yang akan dianalisis, baik melalui kamera maupun galeri perangkat.



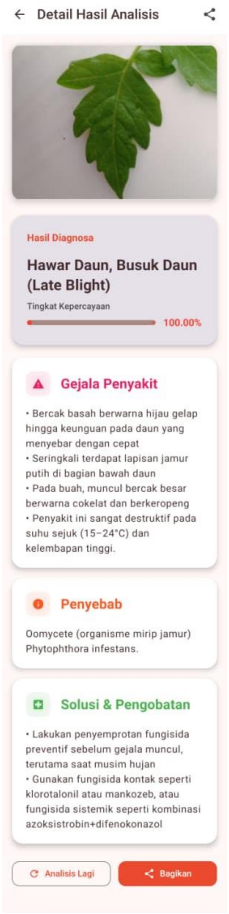
Gambar 9. *Halaman Upload gambar dari CameraX dan Galeri dengan fitur cropping Ucrop*

ISSN: 1978-8126
e-ISSN: 2527-7340

Gambar memperlihatkan proses pemilihan gambar dengan dua opsi: mengambil foto menggunakan CameraX atau memilih file dari galeri. Sebelum dikirim ke server, gambar dipotong menggunakan fitur UCrop agar fokus pada area daun, sekaligus dikompresi otomatis untuk mempercepat proses unggah.

d. Halaman Hasil Deteksi

Halaman ini menampilkan hasil prediksi penyakit berdasarkan model MobileNetV2 yang dijalankan di Azure Cloud.



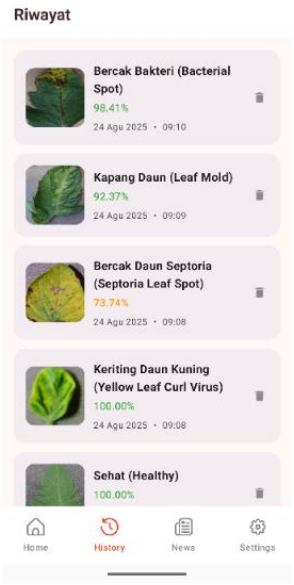
Gambar 10. *Menampilkan prediksi penyakit, confidence score, gejala, penyebab, dan solusi*

Dari gambar terlihat bahwa hasil deteksi disajikan dalam bentuk diagnosis penyakit lengkap dengan tingkat kepercayaan (confidence score), gejala, penyebab, dan rekomendasi penanganan. Pengguna juga dapat menyimpan hasil analisis ke Firestore untuk melihat riwayat di kemudian hari.

e. Halaman Riwayat

Halaman riwayat menampilkan daftar hasil deteksi yang tersimpan di Firebase Firestore berdasarkan waktu analisis.

ISSN: 1978-8126
e-ISSN: 2527-7340

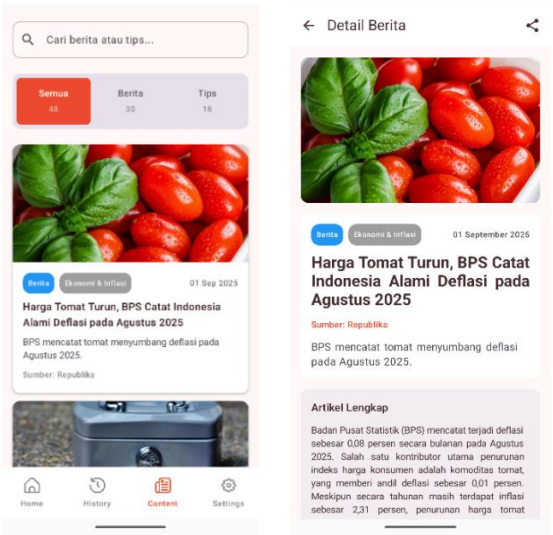


Gambar 11. *Daftar hasil deteksi tersimpan di Firestore dengan opsi hapus*

Gambar menunjukkan bahwa setiap entri riwayat berisi informasi penyakit, tanggal analisis, dan pratinjau gambar yang digunakan. Pengguna juga dapat menghapus data tertentu dengan konfirmasi terlebih dahulu agar tidak terjadi penghapusan secara tidak sengaja.

f. Halaman Konten dan detail konten

Halaman ini menyediakan akses berita dan tips pertanian yang diperoleh dari endpoint /api/content pada backend.



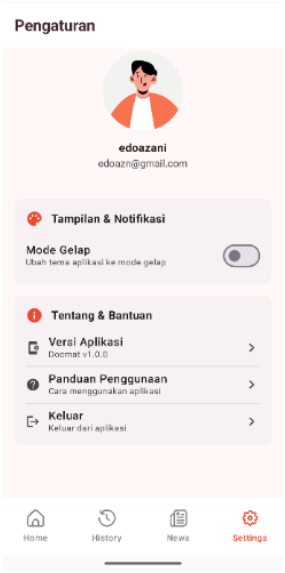
Gambar 12. *Informasi pertanian yang berisi berita dan tips dengan fitur pencarian*

Gambar menampilkan daftar konten pertanian terkini yang dapat dicari berdasarkan kata kunci. Pengguna dapat membaca detail konten untuk mendapatkan wawasan mengenai pencegahan dan pengendalian penyakit tanaman tomat.

ISSN: 1978-8126
e-ISSN: 2527-7340

g. Halaman Pengaturan

Halaman pengaturan memungkinkan pengguna untuk mengubah profil akun, preferensi aplikasi, serta melakukan logout dari sistem.



Gambar 13. *Manajemen akun dan logout*

Dari gambar terlihat bahwa pengguna dapat memperbarui data pribadi seperti nama dan email, serta keluar dari akun dengan aman. Integrasi dengan Firebase memastikan proses *logout* juga menghapus token sesi agar keamanan pengguna tetap terjaga.

3.2.2. Implementasi Backend API

Backend dikembangkan menggunakan FastAPI dengan integrasi model MobileNetV2 untuk klasifikasi penyakit daun tomat. Perubahan signifikan dari versi sebelumnya adalah penggabungan layanan berita dan tips menjadi endpoint konten terpadu.

```
.
└── tomato-api/
    ├── main.py              (entry point FastAPI, definisi endpoint)
    ├── content_service.py   (layanan untuk menggabungkan berita dan tips)
    ├── news_service.py      (sumber data berita)
    ├── tip_service.py       (sumber data tips)
    ├── requirements.txt     (dependensi aplikasi)
    └── .env                 (konfigurasi environment)
```

Gambar 14. *Struktur Backend API*

Penjelasan Singkat Struktur Backend:

1. main.py - entry point aplikasi dan endpoint utama
2. contentservice.py - menggabungkan data dari news dan tips
3. newsservice.py - modul penyedia data berita
4. tipservice.py - modul penyedia tips pertanian
5. schemas - validasi data dengan Pydantic
6. requirements.txt - dependensi aplikasi
7. .env - konfigurasi environment

Endpoint API:

ISSN: 1978-8126
e-ISSN: 2527-7340

Tabel 1. *Tabel endpoint API*

Endpoint	Method	Auth	Deskripsi
/	GET	Tidak	Status API
/predict	POST	Ya	Prediksi penyakit dari gambar
/api/content	GET	Tidak	Daftar konten (berita + tips) pertanian
/api/content/search	GET	Tidak	Pencarian konten berdasarkan kata kunci
/api/content/{content_type}/{content_id}	GET	Tidak	Detail konten tertentu

Contoh *Response Endpoint*:

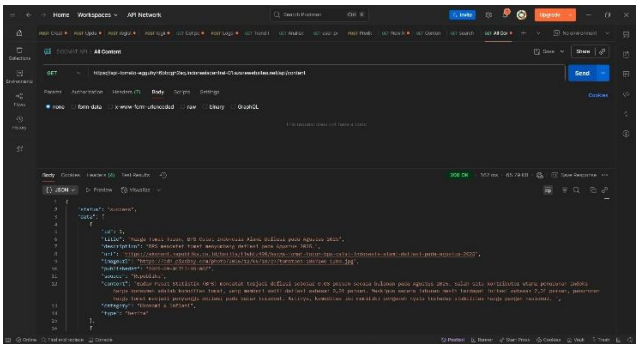
1. Endpoint /predict
   Endpoint ini berfungsi untuk menerima citra daun tomat dari aplikasi Android, kemudian menjalankan proses preprocessing, model inference menggunakan MobileNetV2, dan mengembalikan hasil deteksi penyakit. Endpoint ini bersifat private, sehingga setiap permintaan harus menyertakan Bearer Token dari Firebase Authentication.

```
{
    "status": "success",
    "predict_id": "0f3d9447-2b4e-4970-a667-52a1f3cf2538",
    "timestamp": "2025-08-18T18:57:27.653001Z",
    "model_version": "2.0.0",
    "data": {
        "disease_id": "Bacterial_spot",
        "nama_penyakit": "Bercak Bakteri (Bacterial Spot)",
        "confidence": 0.9840403199195862,
        "confidence_str": "98.40%",
        "gejala": [
            "Bercak kecil, basah, berwarna gelap pada daun, batang, dan buah",
            "Bercak pada daun seringkali memiliki lingkaran kuning di sekelilingnya",
            "Penyakit ini tidak bisa disembuhkan."
        ],
        "penyebab": "Bakteri genus Xanthomonas (X. vesicatoria, X. perforans, dll.).",
        "solusi": [
            "Gunakan benih dan bibit yang bebas penyakit",
            "Lakukan rotasi tanaman 3-4 tahun",
            "Pengendalian berfokus pada pencegahan",
            "Semprotkan bakterisida berbasis tembaga atau kombinasi tembaga-mankozeb segera setelah tanam
atau saat gejala pertama muncul",
            "Aplikasikan kembali sesuai interval pada label produk."
        ],
        "image_url": "https://appku.com/ilustrasi/Bacterial_spot.jpg"
    }
}
```


Gambar 15. *Response sukses dari endpoint predict*

Gambar tersebut menampilkan contoh respons sukses dari server setelah proses inferensi selesai. Respons berisi informasi status, ID prediksi, timestamp, versi model yang digunakan, serta detail hasil deteksi seperti nama penyakit, tingkat kepercayaan (confidence score), gejala, penyebab, dan solusi penanganan. Format respons ini menunjukkan bahwa integrasi antara FastAPI, Firebase Admin SDK, dan TensorFlow telah berjalan dengan baik, memastikan hasil prediksi dapat langsung ditampilkan pada aplikasi Android dan disimpan ke Firestore sebagai riwayat pengguna.

2. Endpoint /api/content
   Endpoint ini menyediakan data konten pertanian berupa berita dan tips yang ditampilkan dalam aplikasi. Endpoint bersifat publik (tanpa autentikasi) dan mengembalikan daftar konten dalam format JSON yang telah disusun berdasarkan kategori dan waktu publikasi.



Gambar 16. *Response Sukses Dari Endpoint /Api/Content*

ISSN: 1978-8126
e-ISSN: 2527-7340

Gambar memperlihatkan struktur respons yang terdiri dari status, total data, dan array objek berisi id, judul, deskripsi, kategori, tanggal publikasi, dan url gambar. Data ini kemudian digunakan aplikasi untuk menampilkan daftar berita serta tips pertanian terkini. Implementasi endpoint ini membuktikan bahwa integrasi antara backend FastAPI dan frontend Android telah berjalan konsisten, dengan format respons yang efisien dan mudah diolah pada lapisan tampilan (*presentation layer*).

3.2.3. Implementasi Model Machine Learning

Model deteksi menggunakan arsitektur MobileNetV2 yang telah dilatih untuk mengklasifikasikan 10 kelas penyakit daun tomat. Dataset yang digunakan berisi 16.012 citra dengan distribusi:

Tabel 2. *Detail dataset citra daun tomat*

No	Kelas Penyakit	Jumlah Citra
1	Septoria Leaf Spot	1.771
2	Healthy	1.591
3	Target Spot	1.404
4	Early Blight	1.000
5	Spider Mites	1.676
6	Late Blight	1.909
7	Leaf Mold	952
8	Bacterial Spot	2.127
9	Mosaic Virus	373
10	Yellow Leaf Curl Virus	3.209

Preprocessing data dilakukan dengan cara mengubah ukuran citra ke resolusi 224×224 piksel agar sesuai input model, menormalkan nilai piksel dari rentang 0–255 menjadi 0–1, serta menerapkan augmentasi data pada set pelatihan berupa rotasi hingga 30°, pergeseran horizontal dan vertikal 20%, shear 20%, zoom 20%, pembalikan horizontal, dan penyesuaian kecerahan.

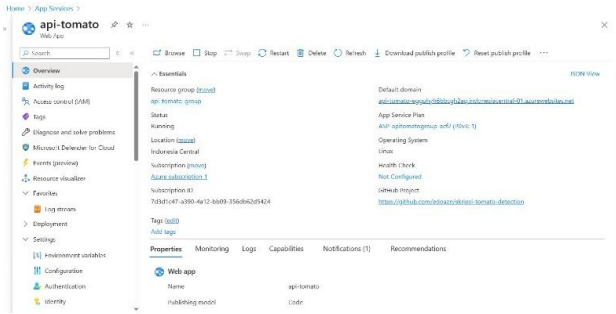
```
# Parameter dasar
batch_size = 32
img_size = (224, 224)

train_datagen = ImageDataGenerator(
    rescale=1./255, rotation_range=30, width_shift_range=0.2, height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest',
    brightness_range=[0.8, 1.2]
)
# Data val dan test TIDAK DIAUGMENTASI, hanya di-rescale
val_test_datagen = ImageDataGenerator(rescale=1./255)

# Generator dataset
train_generator = train_datagen.flow_from_directory(train_dir, target_size=img_size, batch_size=batch_size, class_mode='categorical')
val_generator = val_test_datagen.flow_from_directory(val_dir, target_size=img_size, batch_size=batch_size, class_mode='categorical')
test_generator = val_test_datagen.flow_from_directory(test_dir, target_size=img_size, batch_size=batch_size, class_mode='categorical', shuffle=False)
```

Gambar 17. *Potongan kode preprocessing data*

3.2.4. Integrasi Sistem Cloud-Hybird

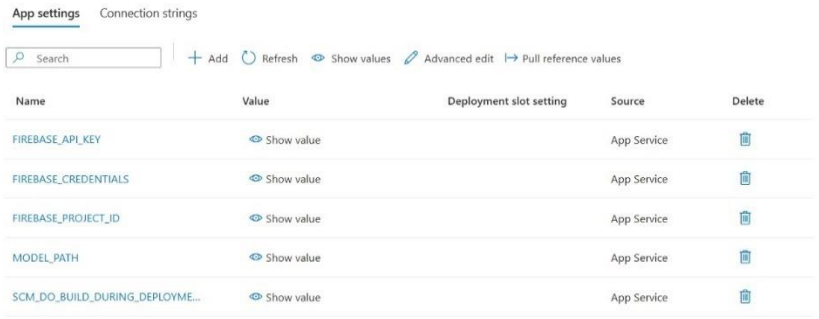
Komponen Cloud Services:

1. Azure App Service - hosting backend FastAPI dengan model TensorFlow
2. Firebase Authentication - autentikasi pengguna dengan ID token JWT
3. Cloud Firestore - penyimpanan riwayat deteksi per pengguna



Gambar 18. *Halaman dashboard azure app service*

ISSN: 1978-8126
e-ISSN: 2527-7340



Gambar 19. *Environment variable azure Web Service*

Alur Integrasi aplikasi:

1. Pengguna login melalui Firebase Authentication dan memperoleh ID token.
2. Gambar diunggah ke endpoint /predict di Azure menggunakan Bearer token.
3. Backend kemudian memverifikasi token dengan Firebase Admin SDK.
4. Gambar diproses melalui tahap preprocessing, dilanjutkan dengan inferensi menggunakan TensorFlow, dan hasilnya dikembalikan dalam bentuk response JSON.
5. Hasil prediksi secara otomatis disimpan ke Firestore berdasarkan userId.
6. Selain itu, pengguna dapat mengakses konten pertanian melalui endpoint /api/content tanpa perlu autentikasi.

Selama implementasi sistem, beberapa tantangan teknis dihadapi terutama pada tahap integrasi layanan *cloud* dan *model machine learning*. Kendala utama muncul pada sinkronisasi antara *Firebase Authentication* dan Azure App Service, di mana validasi token JWT dari *Firebase* harus disesuaikan dengan mekanisme autentikasi di *backend* FastAPI menggunakan *Firebase Admin SDK*. Selain itu, konfigurasi *Cross-Origin Resource Sharing* (CORS) sempat menyebabkan error akses API dari aplikasi Android. Tantangan lain ditemukan pada proses *deployment* model MobileNetV2 di Azure, khususnya waktu inisialisasi model (*model loading time*) yang berdampak pada respons awal prediksi. Untuk mengatasinya, dilakukan optimasi *cache* model dan *warm start* server agar inferensi lebih cepat. Dari sisi aplikasi Android, keterbatasan ukuran file gambar dan koneksi internet pengguna menyebabkan beberapa kasus *timeout* saat unggah citra, sehingga diterapkan kompresi otomatis sebelum pengiriman. Selain itu, sinkronisasi data hasil deteksi antara API Azure dan Firestore juga memerlukan mekanisme *retry* agar tidak terjadi kehilangan data. Tantangan tersebut menjadi pembelajaran penting dalam mengelola sistem *cloud-hybrid* yang menggabungkan *mobile app*, *machine learning*, dan layanan *cloud* secara terpadu.

3.3. Hasil Pengujian Sistem

3.3.1. Pengujian Fungsional (Black Box Testing)

Pengujian dilakukan menggunakan metode *Black Box Testing* untuk memvalidasi 8 fitur utama aplikasi dengan hasil 100% lulus:

Tabel 3. *Pengujian dengan black box testing*

Fitur	Hasil yang Diharapkan	Hasil Uji	Status
Login	Pengguna dapat masuk	Berhasil	Lulus
Register	Pengguna dapat mendaftar	Berhasil	Lulus
Upload Foto	Foto dapat diunggah	Berhasil	Lulus
Prediksi Penyakit	Hasil diagnosis muncul	Berhasil	Lulus
Lihat Riwayat	Riwayat deteksi tampil	Berhasil	Lulus
Notifikasi	Notifikasi terkirim	Berhasil	Lulus
Konten	Konten tampil	Berhasil	Lulus
Pengaturan Akun	Pengaturan tersimpan	Berhasil	Lulus

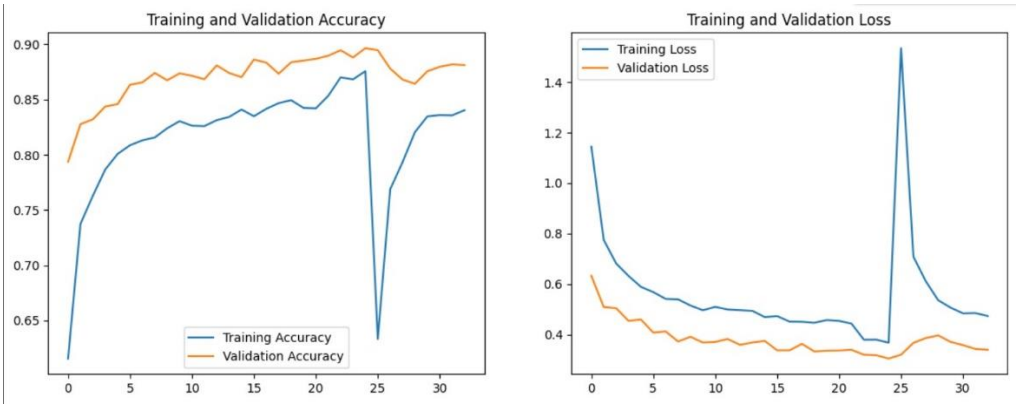
ISSN: 1978-8126
e-ISSN: 2527-7340

3.3.2. Evaluasi Performa Model Machine Learning

Model MobileNetV2 yang diimplementasikan menunjukkan performa sangat baik pada dataset uji Metrik Evaluasi Model:

Tabel 4. Evaluasi performa model

Metrik	Nilai
Akurasi	91,74%
Presisi	91,30%
Recall	92,00%
F1-Score	91,60%



Gambar 20. *Grafik Akurasi dan Loss Pelatihan dan Validasi Model*

Analisis Kurva Training:

1. Akurasi validasi meningkat konsisten dari sekitar 79% (*epoch* 1) hingga sampai 90% (*epoch* 25)
2. Loss validasi menurun stabil hingga 0,30 tanpa indikasi overfitting
3. Model berhasil mencapai target akurasi >85% sesuai spesifikasi penelitian

4. Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Implementasi metodologi *Extreme Programming* dalam pengembangan aplikasi Android deteksi penyakit tanaman tomat berbasis *cloud* terbukti efektif melalui empat tahap utama *planning*, *design*, *coding*, dan *testing* yang dilakukan secara iteratif dan responsif terhadap kebutuhan sistem.
2. Arsitektur *cloud-hybrid* yang mengombinasikan layanan *Firebase Authentication*, *Firestore*, dan *Azure App Service* berhasil diimplementasikan dengan optimal untuk mendukung scalability dan performa sistem yang konsisten dengan response time rata-rata 3 detik pada proses deteksi penyakit.
3. Integrasi model machine learning MobileNetV2 ke dalam aplikasi Android berbasis *cloud* menunjukkan hasil yang memuaskan dengan akurasi mencapai 91,74% dalam mengklasifikasikan 10 kelas penyakit daun tomat, membuktikan efektivitas pendekatan *cloud computing* untuk mengurangi beban komputasi pada perangkat *mobile*.
4. Arsitektur MVVM dengan *Jetpack Compose* memberikan struktur aplikasi yang terorganisir dan mudah dikembangkan, memungkinkan separation of concerns yang baik antar komponen presentation layer, domain layer, dan data layer untuk maintainability yang optimal.
5. Pengujian sistem komprehensif menggunakan *Black Box Testing* menunjukkan bahwa seluruh fitur aplikasi berfungsi sesuai spesifikasi dengan tingkat keberhasilan 100%, mencakup autentikasi pengguna, deteksi penyakit, manajemen riwayat, konten pertanian, dan pengaturan akun.

Daftar Pustaka

[1] A. Abbas *et al.*, "Drones in Plant Disease Assessment, Efficient Monitoring, and Detection: A Way Forward to Smart Agriculture," Jun. 01, 2023, *MDPI*. doi: 10.3390/agronomy13061524.

ISSN: 1978-8126
e-ISSN: 2527-7340

[2] Julvin Saputri Mendrofa, Martirah Warni Zendrato, Nisiyari Halawa, Elias Elwin Zalukhu, and Natalia Kristiani Lase, "Peran Teknologi dalam Meningkatkan Efisiensi Pertanian," *Tumbuhan : Publikasi Ilmu Sosiologi Pertanian Dan Ilmu Kehutanan*, vol. 1, no. 3, pp. 01–12, Sep. 2024, doi: 10.62951/tumbuhan.v1i3.111.

[3] A. N. Mendrofa, N. Gea, and K. Gea, "Pengaruh Pupuk Organik Ampas Kelapa Terhadap Pertumbuhan Tanaman Tomat (Lycopersicum Esculentum Mill)," *Jurnal Sapta Agrica*, vol. 2, pp. 36–49, May 2023, [Online]. Available: https://jurnal.uniraya.ac.id/index.php/Agrica36

[4] A. W. Putri, "Implementasi Artificial Neural Network (Ann) Backpropagation Untuk Klasifikasi Jenis Penyakit Pada Daun Tanaman Tomat," *Jurnal Ilmiah Matematika*, vol. 09, pp. 344–350, Aug. 2021.

[5] M. Astiningrum, P. Prima Arhandi, and N. Aqmarina Ariditya, "Identifikasi Penyakit Pada Daun Tomat Berdasarkan Fitur Warna Dan Tekstur," *JIP (Jurnal Informatika Polinema)* , vol. 6, pp. 47–50, Feb. 2020.

[6] A. C. Anwari, "Klasifikasi Penyakit Tanaman Tomat Melalui Citra Daun Menggunakan Metode Convolutional Neural Network," Online, Ponorogo, 2024.

[7] H. Saputra, I. Stephane, T. A. Sundara, and A. H. Bahri, "Deteksi Penyakit Tanaman Padi Berbasis Android Melalui Pemanfaatan Teachable Machine," *Indonesian Journal of Computer Science*, vol. 13, pp. 1256–1264, Feb. 2024.

[8] U. Khaira, I. Weni, and W. Wilia, "Rancang Bangun Aplikasi Deteksi Penyakit Tanaman Jagung Melalui Citra Daun Berbasis Android Menggunakan Algoritma Convolutional Neural Network," *Jurnal Pepadun*, vol. 5, no. 1, pp. 1–11, 2024.

[9] R. F. Firdaus, L. Novamizanti, and S. A. Wibowo, "Perancangan dan Implementasi Cloud Computing untuk Deteksi Kesegaran Ikan Menggunakan Model Deep Learning YOLOv8 Pada Aplikasi FishQ," *eProceedings of Engineering*, vol. 11, no. 4, pp. 2870–2880, Aug. 2024.

[10] D. C. Wijaya, L. Novamizanti, A. Fakultas, and T. Elektro, "Implementasi Machine Learning pada Google Cloud Platform untuk Mengidentifikasi Tingkat Kualitas Ikan Menggunakan Aplikasi Android," *e-Proceeding of Engineering*, pp. 526–531, 2024, [Online]. Available: www.pulumi.com

[11] V. Y. P. Ardhana, "Sistem Informasi Kebencanaan Berbasis Android Menggunakan Metode Extreme Programming," *Jambura Journal of Informatics*, vol. 4, no. 2, pp. 61–69, Nov. 2022, doi: 10.37905/jji.v4i2.16057.