PENGEMBANGAN LAYANAN BACKEND BERBASIS FLASK UNTUK APLIKASI ANDROID DETEKSI PENYAKIT DAUN DAN JENIS DURIAN MENGGUNAKAN DIGITAL OCEAN DENGAN METODE KANBAN

(Skripsi)

Oleh

ARBIAN ALEX PRITAMA

2155061013

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2026

PENGEMBANGAN LAYANAN BACKEND BERBASIS FLASK UNTUK APLIKASI ANDROID DETEKSI PENYAKIT DAUN DAN JENIS DURIAN MENGGUNAKAN DIGITAL OCEAN DENGAN METODE KANBAN

Oleh

ARBIAN ALEX PRITAMA

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2026

ABSTRAK

PENGEMBANGAN LAYANAN BACKEND BERBASIS FLASK UNTUK APLIKASI ANDROID DETEKSI PENYAKIT DAUN DAN JENIS DURIAN MENGGUNAKAN DIGITAL OCEAN DENGAN METODE KANBAN

Oleh

ARBIAN ALEX PRITAMA

Durian merupakan salah satu komoditas pertanian unggulan di Indonesia, namun proses identifikasi penyakit daun serta klasifikasi varietas buah durian masih banyak dilakukan secara manual.Tujuan dari penelitian ini adalah untuk mengembangkan layanan *backend* berbasis Flask yang mampu menerima input citra dari aplikasi Android, melakukan klasifikasi jenis durian dan dekteksi penyakit pada daun durian, serta menyediakan riwayat hasil pemindaian. Sistem dibangun dalam bentuk REST API yang menangani dua fitur utama, yaitu klasifikasi penyakit daun durian dan identifikasi varietas buah durian. Proses pengerjaan sistem ini menggunakan metode kanban yang dibagi menjadi 4 tahap yaitu *backlog*, *To-do*, *Work in progresss* dan *Done* dengan 16 *task* dalam kanban *board* yang berdurasi waktu pengerjaan *task* 52 hari waktu efektif. *Backend* terhubung dengan basis data untuk menyimpan hasil prediksi sebagai riwayat pengguna. *Deployment* sistem dilakukan pada layanan *cloud DigitalOcean* agar REST API dapat diakses secara *online* oleh aplikasi *mobile*. Hasil pengujian menunjukkan bahwa sistem mampu memberikan keluaran prediksi melalui *endpoint* API serta berjalan stabil pada lingkungan *deployment*. Penelitian ini menghasilkan sistem *backend* dengan 7 *endpoint* API dengan 2 *endpoint* utama yaitu */predict/disease* dan */predict/variety* yang sudah diimplementasikan kedalam sistem aplikasi Duri Pintar.

Kata kunci: Durian, Flask, REST API, *DigitalOcean, Deployment.*

ABSTRACT

FLASK-BASED BACKEND SERVICE DEVELOPMENT FOR ANDROID APPLICATIONS TO DETECTION OF LEAF DISEASES AND DURIAN TYPES USING DIGITAL OCEAN WITH THE KANBAN METHOD

By

ARBIAN ALEX PRITAMA

Durian is one of Indonesia's leading agricultural commodities; however, the process of identifying leaf diseases and classifying durian varieties is still largely performed manually. The purpose of this study is to develop a Flask-based backend service capable of receiving image input from an Android application, performing durian variety classification and durian leaf disease detection, and providing a history of scan results. The system is developed in the form of a REST API that handles two main features, namely durian leaf disease classification and durian variety identification. The development process applies the Kanban method, which is divided into four stages: backlog, to-do, work in progress, and done. A total of 16 tasks were managed in the Kanban board with an effective development duration of 52 days. The backend is integrated with a database to store prediction results as user history. The system is deployed on the DigitalOcean cloud platform so that the REST API can be accessed online by the mobile application. Testing results indicate that the system is able to generate prediction outputs through API endpoints and operates stably in the deployment environment. This study produces a backend system consisting of seven API endpoints, with two main endpoints, namely /predict/disease and /predict/variety, which have been successfully implemented in the Duri Pintar application.

Keywords: Durian, Flask, REST API, DigitalOcean, Deployment.

Judul Skripsi	:	PENGEMBANGAN LAYANAN BACKEND BERBASIS FLASK UNTUK APLIKASI ANDROID DETEKSI PENYAKIT DAUN DAN JENIS DURIAN MENGGUNAKAN DIGITAL OCEAN DENGAN METODE KANBAN
Nama Mahasiswa	:	Arbian Alex Pritama
Nomor Pokok Mahasiswa	:	2155061013
Program Studi	:	Teknik Informatika
Jurusan	:	Teknik Elektro
Fakultas	:	Teknik

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Utama	Pembimbing Pendamping
Ir. Gigih Forda Nama, S.T., M.T.I., IPM	Ir. Trisya Septiana, S. T., M.T., IPM
NIP. 198307122008121003	NIP. 199009212019032025

2. Mengetahui

Ketua Jurusan	Ketua Program Studi
Teknik Elektro	Teknik Informatika
Herlinawati, S.T., M.T.	Yessi Mulyani, S.T., M.T.
NIP. 197103141999032001	NIP. 197312262000122001

MENGESAHKAN

1. Tim Penguji

Ketua : **Ir. Gigih Forda Nama, S.T., M.T.I., IPM** .....

Sekretaris : **Ir. Trisya Septiana, S. T., M.T., IPM** .....

Penguji : **Dr. Eng. Ir. Mardiana, S.T., M.T., IPM** .....

2. Dekan Fakultas Teknik

Dr. H. Ahmad Herison, S.T., M.T.

NIP. 196910302000031001

Tanggal Lulus Ujian Skripsi : 10 Februari 2026

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi berjudul "Pengembangan Layanan Backend Berbasis Flask Untuk Aplikasi Android Deteksi Penyakit Daun Dan Jenis Durian Menggunakan Digital Ocean Dengan Metode Kanban" sepenuhnya merupakan hasil karya saya sendiri. Apabila di kemudian hari terbukti pernyataan ini tidak benar, saya siap menerima sanksi sesuai dengan ketentuan hukum yang berlaku.

Bandar Lampung, 10 Februari 2026

Penulis,

Arbian Alex Pritama

NPM. 2155061013

RIWAYAT HIDUP

Penulis dilahirkan di Bandar Lampung pada tanggal 24 Oktober 2003, sebagai anak pertama dari pasangan Bapak Rizal dan Ibu Isti. Penulis menyelesaikan pendidikan formal di SD Muhammadiyah pada tahun 2015, kemudian melanjutkan ke SMP Xaverius dan lulus pada tahun 2018, serta menamatkan pendidikan menengah atas di SMA Negeri 1 Tanjung Sakti Pumi pada tahun 2020. Pada tahun 2021, penulis diterima sebagai mahasiswa Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung melalui jalur seleksi SMMPTN barat. Selama masa perkuliahan, penulis aktif berpartisipasi dalam berbagai kegiatan, antara lain:

1. Mengikuti kegiatan Studi Independen Bersertifikat dari Kementerian Pendidikan dan Budaya di mitra Bangkit Akademi pada tahun 2023.
2. Menjadi anggota Himpunan Mahasiswa Teknik Elektro pada tahun 2022-2023.

MOTTO

"Berfokus pada gaya hidup sehat, konsistensi, dan kemandirian."

(Dr. Tirta Mandira Hudhi)

"*I have not failed. I've just found 10,000 ways that won't work.*"

(Thomas Edison)

"Kesempatan tidak datang begitu saja, Kamu yang menciptakannya."

(Penulis)

PERSEMBAHAN

Dengan penuh rasa syukur kepada Allah SWT atas limpahan rahmat, karunia, dan petunjuk-Nya, karya sederhana ini saya persembahkan kepada:

Kedua orang tua tercinta, yang telah menjadi sumber kekuatan, kasih sayang, dan doa dalam setiap langkah hidup saya.

Keluarga besar yang selalu memberi semangat dan dukungan tanpa henti.

Sahabat-sahabat seperjuangan yang senantiasa hadir dalam suka dan duka.

Serta almamater kebanggaan, Universitas Lampung, tempat saya menimba ilmu dan pengalaman berharga.

SANWACANA

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul **"**Pengembangan Layanan Backend Berbasis Flask Untuk Aplikasi Android Deteksi Penyakit Daun Dan Jenis Durian Menggunakan Digital Ocean Dengan Metode Kanban**"** dengan baik.

Penyusunan skripsi ini tidak terlepas dari dukungan, bimbingan, serta bantuan berbagai pihak yang telah memberikan kontribusi, baik dalam bentuk moril maupun materil. Oleh karena itu, dengan penuh rasa hormat dan ketulusan, penulis menyampaikan ucapan terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Kedua orang tua tercinta beserta keluarga besar yang senantiasa memberikan doa, kasih sayang, dukungan, serta motivasi yang tiada henti kepada penulis, baik secara moril maupun spiritual, sehingga penulis mampu menyelesaikan studi dan penelitian ini dengan penuh semangat.;
2. Bapak Dr. Hi. Ahmad Herison, S.T., M.T. selaku Dekan Fakultas Teknik, Universitas Lampung.;
3. Ibu Herlinawati, S.T., M.T., selaku Ketua Jurusan Teknik Elektro Universitas Lampung.;
4. Ibu Yessi Mulyani, S.T., M.T., selaku Ketua Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung, yang telah memberikan kesempatan, bimbingan, serta dorongan kepada penulis dalam menyelesaikan skripsi ini.;
5. Bapak Puput Budi Wintoro, S.Kom., M.T.I., selaku Dosen Pembimbing Akademik, yang telah memberikan bimbingan, perhatian, serta motivasi selama penulis menjalani masa perkuliahan.;

6. Bapak Ir. Gigih Forda Nama, S.T., M.T.I., IPM, selaku Dosen Pembimbing Utama, yang dengan sabar, telaten, dan penuh perhatian telah memberikan arahan, masukan, serta ilmu yang sangat berharga selama proses penelitian dan penulisan skripsi ini.;
7. Ibu Ir. Trisya Septiana, S. T., M.T., IPM, selaku Dosen Pembimbing Pendamping, yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, saran, serta dukungan dalam penyempurnaan penelitian ini.;
8. Ibu Dr. Eng. Ir. Mardiana, S.T., M.T., IPM, selaku Dosen Penguji, yang telah memberikan kritik, saran, dan masukan yang membangun demi penyempurnaan hasil penelitian ini.;
9. Seluruh Dosen Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung, yang telah memberikan ilmu, wawasan, dan pengalaman berharga selama masa perkuliahan.;
10. Rekan satu tim penelitian, Muchammad Raja Haikal Fiaugustian, Farhat Febrianto, Muhammad Rafi Rizanda, atas kerja sama yang solid, semangat kebersamaan, serta kontribusinya dalam pengembangan Aplikasi Duri Pintar ini.;
11. Teman-teman dalam grup “Info Loker”, serta seluruh sahabat Teknik Informatika 2021 yang tidak dapat disebutkan satu per satu, atas kebersamaan, dukungan, dan semangat yang diberikan selama masa perkuliahan hingga penyusunan skripsi ini.;
12. Kepada nama-nama ini : Muhammad Bintang Khadafi, Vonny Tri Agusta, Meisya Amelia, Khalidah Rosa Amaliah, Risna Purnama Sari, Sherina Agtha Bellen yang memberi bantuan dan dukungan dalam mengerjakan skripsi ini.;
13. Kepada Tetangga kos yang bernama Alida Shidqiya Naifa Ulmuflikhun ucapan terimakasih yang sebesar-besarnya karena telah berperan penting memberi inspirasi serta memberi semangat kepada penulis dalam proses penyusunan skripsi.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, sehingga kritik dan saran yang membangun sangat diharapkan. Semoga karya ini dapat memberikan manfaat serta menjadi referensi bagi pengembangan penelitian di bidang Rekayasa

Perangkat Lunak dan komputasi *backend*. Penulis juga berharap seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyusunan skripsi ini senantiasa mendapatkan limpahan rahmat dan balasan kebaikan dari Allah SWT.

Bandar Lampung, 10 Februari 2026
Penulis,

Arbian Alex Pritama
NPM. 2155061013

DAFTAR ISI

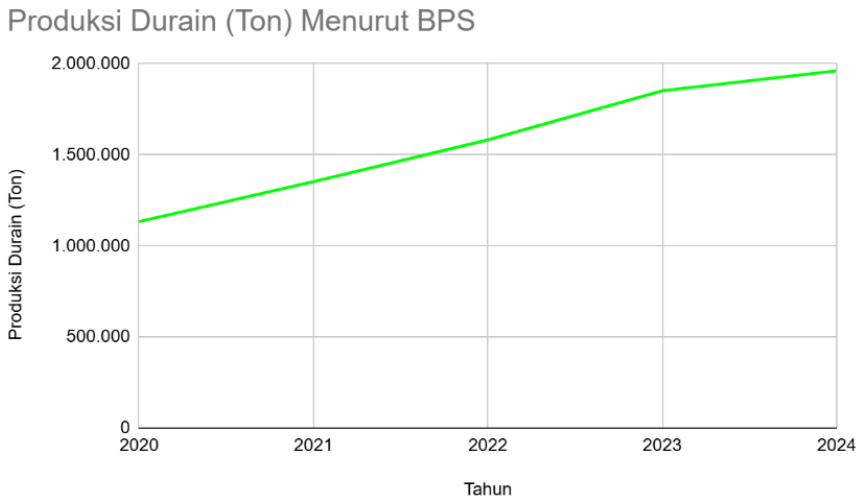
DAFTAR TABEL

DAFTAR GAMBAR

I. PENDAHULUAN

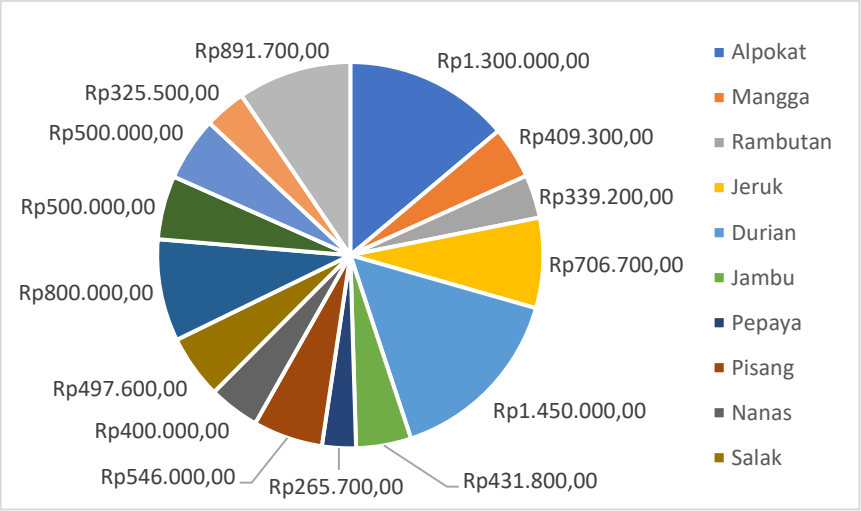
1.1 Latar Belakang

Durian atau disebut dalam bahasa latin yaitu *Durio Zubethinus Murr* merupakan salah satu jenis buah tropis yang tergolong dalam famili *Bombacaceae* yang memiliki nilai ekonomis tinggi [1]. Berdasarkan data yang dirilis oleh Badan Pusat Statistik, produksi durian di Indonesia mengalami peningkatan yang konsisten dari tahun ke tahun. Pada tahun 2020, total produksi durian tercatat sebesar 1.133.194 ton. Jumlah tersebut meningkat menjadi 1.352.957 ton pada tahun 2021, kemudian naik menjadi 1.582.171 ton pada tahun 2022. Tren kenaikan ini berlanjut pada tahun 2023 dengan total produksi sebesar 1.852.045 ton, dan pada tahun 2024 mencapai sekitar 1.961.486 ton. Data ini menunjukkan adanya pertumbuhan yang signifikan dalam sektor produksi durian selama lima tahun terakhir.



Gambar 1.1.Peningkatan Produksi Durian Di Indonesia [2]

Produksi durian menunjukkan dampak ekonomi yang signifikan bagi perekonomian di Indonesia. Harga jual durian yang relatif tinggi dibandingkan dengan komoditas buah lainnya menjadikannya sebagai salah satu peluang agribisnis yang potensial dalam meningkatkan pendapatan petani. Tingkat produksi durian yang terus meningkat setiap tahunnya membuka peluang untuk ekspansi ke pasar internasional.



Gambar 1.2. Harga Jual Durian (Kw) dengan Komoditas Lain [3]

Penerapan kecerdasan buatan (*Artificial Intelligence*/AI) dalam bidang pertanian telah menunjukkan potensi besar dalam meningkatkan efisiensi dan akurasi deteksi penyakit tanaman. Salah satu pendekatan yang banyak digunakan adalah *Convolutional Neural Network* (CNN), yang efektif dalam mengenali pola visual pada gambar daun atau buah tanaman [4].

Menerapkan model AI seperti CNN dalam aplikasi *mobile*, diperlukan sistem *backend* yang handal. *Backend* berfungsi sebagai penghubung antara aplikasi *frontend* dan model AI, serta mengelola penyimpanan data dan komunikasi antar komponen sistem. *Flask*, sebuah framework web berbasis *Python*, sering digunakan untuk membangun *REST API* yang ringan dan fleksibel, memungkinkan integrasi yang efisien dengan model AI dan basis data seperti *PostgreSQL*.

PostgreSQL dipilih sebagai basis data karena memiliki fitur yang lengkap, stabil, serta mendukung pengembangan sistem secara efisien., serta dukungannya terhadap berbagai jenis data, termasuk data spasial dan *.JSON*. Integrasi *Flask* dengan *PostgreSQL* memungkinkan pengelolaan data pengguna, hasil klasifikasi, dan histori pemindaian secara efisien dan terstruktur.

Sistem *backend* yang menjadi tulang punggung dari aplikasi mobile pendeteksi penyakit daun dan klasifikasi jenis tanaman durian. Fokus utama penelitian adalah

pada perancangan dan pengembangan *backend* menggunakan *framework Flask*, yang bertugas menangani komunikasi antara aplikasi *mobile* dengan model *machine learning* serta basis data. *Backend* ini juga berfungsi sebagai pengelola data hasil pemindaian daun durian dan riwayat pengguna, serta di-*deploy* menggunakan *Digital Ocean* agar dapat diakses secara publik.

1.2 Rumusan Masalah

Adapun rumusan masalah dari penelitian ini, dengan fokus pada pengembangan *backend* aplikasi adalah sebagai berikut :

1. Bagaimana merancang *backend* berbasis *Flask* untuk klasifikasi penyakit daun dan jenis buah durian menggunakan model *machine learning*?
2. Bagaimana membangun REST API yang mendukung *input* gambar, integrasi model ML, dan penyimpanan data ke PostgreSQL?
3. Bagaimana cara melakukan *deployment backend* ke *Digital Ocean* untuk diakses oleh aplikasi *mobile*?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini difokuskan pada pembangunan sistem *backend* dari aplikasi deteksi durian adalah sebagai berikut:

1. Mengembangkan layanan *backend* berbasis *Flask* yang mampu menerima input gambar dari aplikasi Android, melakukan klasifikasi menggunakan model *machine learning*, dan mengembalikan hasil riwayat nya.
2. Membangun *REST API* yang dapat menangani dua fitur utama: klasifikasi penyakit daun durian dan identifikasi jenis buah durian.
3. Menerapkan *deployment backend* ke *Digital Ocean* agar dapat diakses secara online oleh aplikasi *mobile*.

1.4 Manfaat Penelitian

Adapun manfaat penelitian ini adalah sebagai berikut:

1. Bagi petani, penelitian ini dapat membantu dalam mengidentifikasi penyakit daun pada tanaman durian secara cepat dan akurat.
2. Bagi masyarakat umum atau pengguna awam, sistem yang dikembangkan diharapkan dapat menjadi alat bantu visual yang mudah digunakan untuk mengenali jenis penyakit pada daun tanaman durian, meskipun tanpa pengetahuan teknis di bidang pertanian atau penyakit tanaman.

1.5 Batasan Masalah

Adapun batasan masalah pada penelitian ini ditentukan untuk menjaga fokus pada aspek pengembangan backend sistem adalah sebagai berikut:

1. Deteksi dilakukan berdasarkan gambar daun, sedangkan identifikasi jenis durian dilakukan berdasarkan gambar buah.
2. Backend dikembangkan menggunakan Flask dengan basis data PostgreSQL, dan proses deployment dilakukan melalui layanan DigitalOcean.
3. Sistem ini mengimplementasikan tiga endpoint REST API, yaitu klasifikasi penyakit daun, klasifikasi jenis durian, dan integrasi API cuaca.
4. Penelitian ini tidak membahas aspek keamanan, performa sistem, maupun evaluasi akurasi deteksi dan identifikasi karena model yang digunakan merupakan hasil pengembangan pihak lain.
5. Fokus penelitian hanya pada implementasi serta integrasi fitur backend yang mendukung fungsionalitas utama aplikasi.

1.6 Sistematika Penulisan Laporan

Adapun sistematika yang digunakan dalam penulisan penelitian ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Bab ini berisi latar belakang penelitian yang menjelaskan terkait konteks penelitian tentang latar belakang dalam pengembangan sistem klasifikasi jenis penyakit pada daun durian, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II : LANDASAN TEORI

Bab ini berisi teori-teori dasar yang mendukung penelitian dan konsep deteksi penyakit tanaman, serta penelitian-penelitian sebelumnya yang terkait dengan penelitian ini.

BAB III : METODOLOGI PENELITIAN

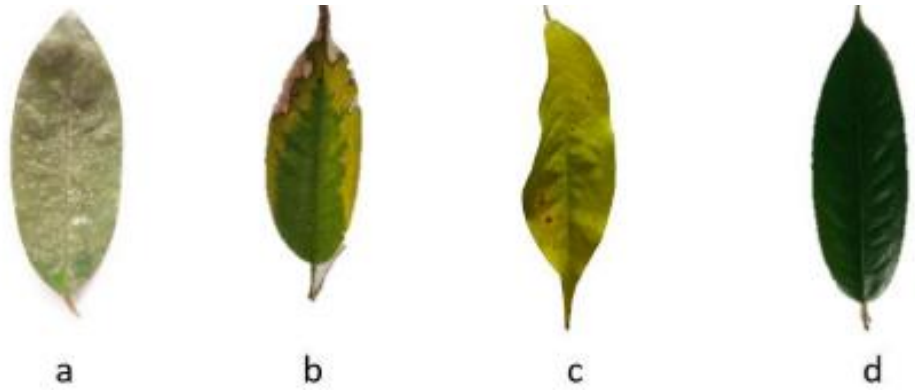
Bab ini berisi metode yang digunakan dalam mengembangkan aplikasi, mulai dari waktu penelitian, tempat penelitian, alat penelitian serta metode dan tahap yang dilakukan dalam penelitian ini.

II.TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Penyakit Pada Daun Durian

Tanaman durian (*Durio Zibethinus*) rentan terhadap berbagai penyakit yang menyerang bagian daun, yang secara signifikan dapat mengganggu produktivitas dan kesehatan tanaman. Salah satu masalah yang umum dijumpai adalah infeksi alga seperti *Cephaleuros virescens*, yang menyebabkan bercak kehijauan hingga oranye pada permukaan daun, mengganggu fotosintesis, dan berpotensi menyebabkan kerontokan daun. Selain itu, serangan hama seperti *Allocaridara malayensis* (kutu loncat) juga menjadi ancaman serius dengan gejala daun menggulung, menguning, serta munculnya embun madu yang mendorong pertumbuhan jamur jelaga [5].



Gambar 2.1 Contoh Gambar Daun Durian

(*Source*: https://chanelmuslim.com/healthy/manfaat-utama-daun-durian-untuk-kesehatan-tubuh)

Dalam suatu penelitian menemukan bahwa infeksi laten oleh *P. durionis* dapat terjadi pada jaringan daun dan bunga yang tampak sehat, menunjukkan potensi penyebaran tanpa gejala yang jelas [6]. Selain itu, penyakit *leaf blight* yang disebabkan oleh *Rhizoctonia solani subgroup* AG 1-ID juga telah dilaporkan pada tanaman durian di Vietnam. Gejala penyakit ini meliputi lesi besar berwarna cokelat pucat dengan tepi tidak beraturan, yang dapat menyebabkan layu dan kematian daun [7].

Gambar 2.2 Contoh Penyakit *Leaf Blight* Pada Durian

(*Source*: https://sensing.konicaminolta.asia/id/detecting-durian-leaf-disease-leaf-blight-leaf-spot-anthracnose-with-hyperspectral-imaging/)

2.1.2 Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat *open source*, dikembangkan pertama kali oleh Guido van Rossum pada akhir tahun 1980-an dan dirilis secara resmi pada tahun 1991. *Python* digunakan di berbagai bidang, seperti pengembangan perangkat lunak, kecerdasan buatan (AI), pengembangan web, dan analisis data. *Python* dirancang dengan filosofi penekanan pada keterbacaan kode dan sintaks yang bersih sehingga mudah dipelajari oleh pemula, namun juga cukup kuat untuk digunakan dalam pengembangan aplikasi yang kompleks [8].

Gambar 2.3 Logo Bahasa Pemrograman Python

(*Source*: https://id.wikipedia.org/wiki/Berkas:Python_logo_and_wordmark.svg)

Dalam pengembangan perangkat lunak modern, *Python* telah menjadi salah satu bahasa yang paling populer karena fleksibilitasnya yang tinggi dan dukungan pustaka (*library*) yang sangat luas. Salah satu kekuatan utama *Python* terletak pada kemampuannya dalam menangani berbagai kebutuhan pengembangan seperti automasi, pemrosesan data, pengembangan web, hingga implementasi kecerdasan buatan dan *machine learning* [9].

Pada penelitian ini, *Python* digunakan sebagai bahasa utama dalam pengembangan sisi *backend* aplikasi, khususnya untuk membangun REST API menggunakan *framework Flask*. Selain itu, *Python* juga digunakan untuk memuat dan menjalankan model *machine learning* dalam format *.json* yang bertugas untuk memproses gambar daun durian dan menghasilkan hasil prediksi. Dengan dukungan pustaka seperti *TensorFlow*, dan *NumPy*,

2.1.3 Flask Framework

Flask adalah salah satu *web framework* berbasis *Python* yang bersifat *mikro* dan *open source*. Framework ini pertama kali diperkenalkan oleh Armin Ronacher pada tahun 2010 dan menjadi sangat populer di kalangan pengembang karena kesederhanaan serta fleksibilitasnya [10]. *Flask* dirancang untuk memberikan kontrol penuh kepada pengembang terhadap struktur dan arsitektur aplikasi *web* yang dibangun, tanpa terlalu banyak "aturan main" seperti yang dimiliki oleh *framework* yang lebih besar seperti *Django* [11].

Gambar 2.4 Logo Flask Framework

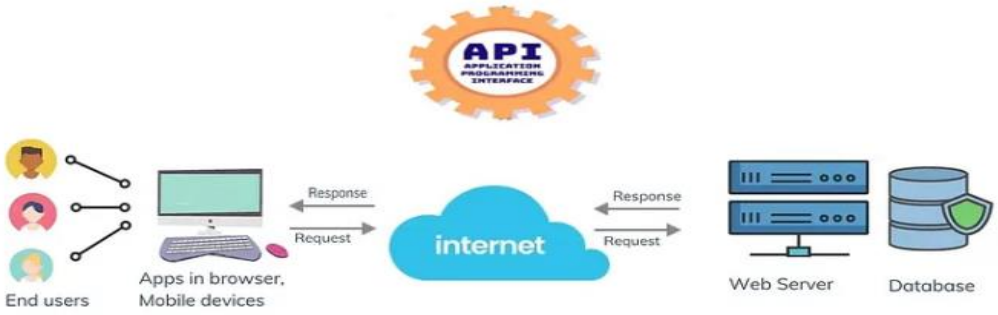
(*Source*: https://id.wikipedia.org/wiki/Berkas:Flask_logo.svg)

Flask tidak menyertakan komponen seperti ORM (*Object-Relational Mapping*), validasi form, atau otentikasi secara default. Namun, kekurangan tersebut justru menjadi keunggulan dalam proyek tertentu karena pengembang dapat memilih dan menambahkan komponen sesuai kebutuhan proyeknya melalui ekstensi yang tersedia. *Flask* sangat cocok untuk pengembangan aplikasi kecil hingga menengah, API berbasis REST, serta prototipe aplikasi yang membutuhkan waktu pengembangan yang cepat, meskipun tidak secepat *Express.js* dan *Laravel* [12].

Pada penelitian ini, *Flask* digunakan untuk membangun sistem *backend* dari aplikasi deteksi tanaman durian. *Backend* tersebut bertanggung jawab untuk menerima input gambar dari pengguna melalui aplikasi mobile, kemudian memproses gambar tersebut menggunakan model machine learning yang telah dilatih sebelumnya, dan akhirnya menyimpan hasil prediksi ke *database* agar dapat diakses kembali oleh pengguna.

2.1.4 API (Application Programming Interface)

Application Programming Interface (API) adalah sekumpulan definisi dan protokol yang memungkinkan satu aplikasi berinteraksi dengan aplikasi lainnya. API bertindak sebagai jembatan antara dua sistem perangkat lunak yang berbeda, sehingga pengembang dapat menggunakan fungsionalitas tertentu tanpa harus membangun semuanya dari awal. Dalam konteks pengembangan *web*, API sering digunakan untuk pertukaran data antara *frontend* dan *backend* melalui protokol HTTP, biasanya dalam format *JSON* atau *XML*. Penggunaan API mempermudah integrasi antar sistem serta mempercepat proses pengembangan perangkat lunak [13].



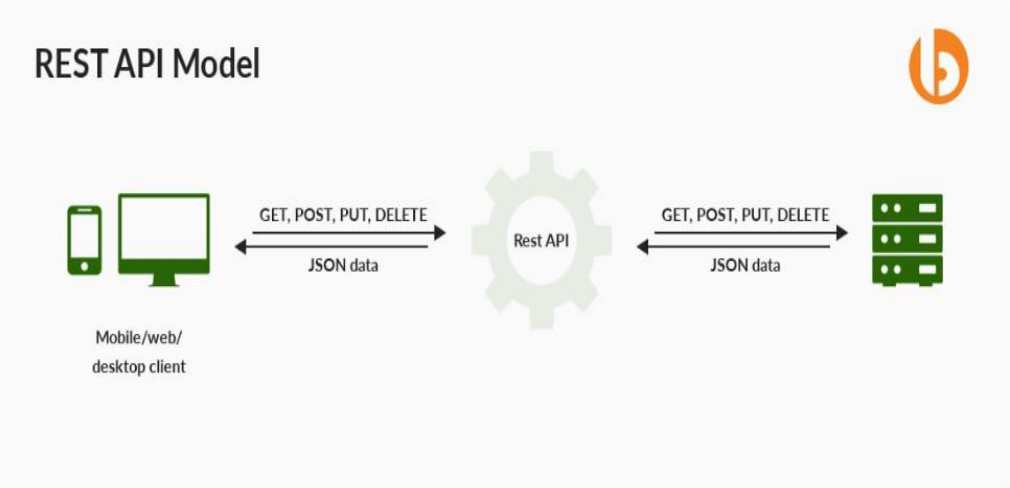
Gambar 2.5 Struktur Kerja API

(*Source*: https://www.goldenfast.net/blog/api-adalah/)

Penggunaan API sangat penting dalam pengembangan perangkat lunak modern karena memungkinkan integrasi layanan pihak ketiga seperti sistem pembayaran, autentikasi, atau layanan peta. Selain itu, API juga digunakan dalam komunikasi antara *frontend* dan *backend*, di mana *frontend* akan mengirimkan permintaan data dan backend akan merespons dengan data yang dibutuhkan. Hal ini meningkatkan efisiensi, keamanan, dan skalabilitas dalam pengembangan aplikasi.

2.1.5 RESTful API

RESTful API (*Representational State Transfer Application Programming Interface*) adalah suatu pendekatan dalam merancang arsitektur antarmuka pemrograman aplikasi (API) yang berbasis pada protokol HTTP dan prinsip REST. REST sendiri merupakan arsitektur komunikasi yang dikenalkan oleh Roy Fielding pada tahun 2000 melalui disertasinya, dan telah menjadi salah satu pendekatan paling populer dalam pengembangan layanan web modern karena kesederhanaannya, skalabilitasnya, dan kemudahan dalam implementasi [14].



Gambar 2.6 Struktur REST API

(*Source* : https://course-net.com/blog/rest-api-adalah-pengertian-fungsi-dan-cara-kerja/)

RESTful API memungkinkan dua sistem berbeda seperti *frontend* dan *backend* atau *client* dan *server* untuk berkomunikasi dan bertukar data secara efisien melalui permintaan HTTP standar, seperti GET, POST, PUT, dan DELETE. Data yang dikirim dan diterima biasanya menggunakan format yang ringan dan mudah dibaca, seperti JSON (*JavaScript Object Notation*) [15].

Dalam penelitian ini, RESTful API digunakan sebagai penghubung antara aplikasi mobile berbasis Kotlin (sebagai *client*) dan *backend* berbasis *Flask* (sebagai *server*). API ini bertugas untuk menerima data gambar yang diunggah oleh pengguna, mengolah gambar menggunakan model *machine learning*, dan mengembalikan hasil prediksi dalam format *JSON*. Penggunaan RESTful API dalam sistem ini mempermudah integrasi antara *backend Flask* dan aplikasi mobile, serta mendukung pengembangan sistem yang fleksibel, modular, dan mudah untuk diperluas di masa depan.

2.1.6 Machine Learning Pada Backend

Machine Learning (ML) adalah cabang dari kecerdasan buatan (*Artificial Intelligence*) yang memungkinkan sistem komputer untuk belajar dari data tanpa harus diprogram secara eksplisit. Dalam konteks pengembangan sistem informasi

atau aplikasi modern, machine learning dapat digunakan untuk membuat sistem yang mampu mengenali pola, membuat prediksi, atau mengambil keputusan berdasarkan data yang dianalisis. Model *machine learning* yang digunakan pada *backend* dikembangkan menggunakan *framework* seperti *TensorFlow* atau *Keras* dan diekspor ke dalam format *.json* (umumnya model *TensorFlow.js* atau model konfigurasi terstruktur). Format *.json* ini menyimpan arsitektur model serta bobot (*weights*) dan konfigurasi yang diperlukan untuk memuat model kembali ke dalam memori komputasi.

Model *machine learning* umumnya terdiri dari beberapa lapisan (*layers*) seperti *convolutional neural network* (CNN), *activation function*, dan *dense layer*. Arsitektur ini ditentukan saat model dilatih, kemudian disimpan dalam format *.json*. *Backend* menggunakan pustaka seperti *TensorFlow*, *TensorFlow Lite*, *Keras*, atau *numpy* untuk memproses model tersebut [16].

Penerapan *machine learning* pada *backend* merupakan pendekatan efektif untuk menyelesaikan masalah klasifikasi gambar secara otomatis. Dalam penelitian ini, penggunaan model ML untuk mendeteksi penyakit dan mengenali varietas durian menjadi fitur utama aplikasi. Integrasi model melalui *backend Flask* memastikan pemrosesan yang terpusat, aman, dan fleksibel dalam pengelolaan, sehingga meningkatkan kualitas dan efisiensi sistem secara keseluruhan.

2.1.7 Digital Ocean

Digital Ocean merupakan layanan *Infrastructure as a Service* (IaaS) dari *Digital Ocean* yang menyediakan mesin virtual (VM) untuk menjalankan berbagai jenis aplikasi, termasuk aplikasi backend. Dalam proyek ini, *Digital Ocean* digunakan untuk mendeploy backend berbasis *Flask* yang terhubung dengan model machine learning dan berfungsi sebagai jembatan antara aplikasi mobile dan model deteksi penyakit atau klasifikasi jenis durian [17].

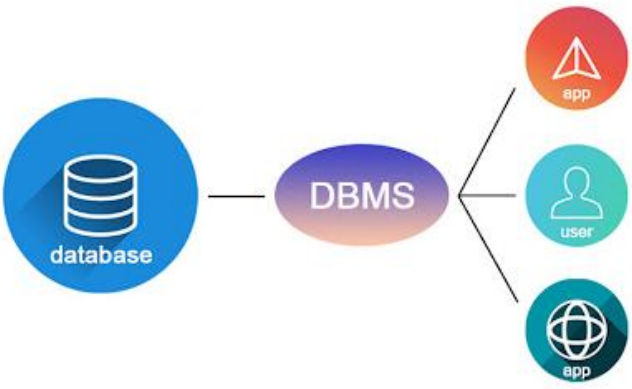
Gambar 2.7. Digital Ocean

(*Source* : https://commons.wikimedia.org/wiki/File:DigitalOcean_logo.svg)

Penggunaan *Digital Ocean* memungkinkan *backend Flask* dapat diakses secara online melalui IP publik atau domain yang terhubung. Hal ini sangat membantu karena aplikasi mobile membutuhkan *endpoint* API yang selalu tersedia secara daring. *Digital Ocean* juga memungkinkan pengguna untuk memilih spesifikasi VM sesuai kebutuhan, serta memberikan fleksibilitas dalam mengatur *firewall*, *storage*, dan konfigurasi sistem operasi.

2.1.8 Basis Data

Basis data adalah sistem penyimpanan terstruktur yang memungkinkan data disimpan, diakses, dan dimanipulasi secara efisien. Dalam konteks pengembangan perangkat lunak, basis data digunakan untuk menyimpan informasi penting seperti data pengguna, produk, transaksi, dan log aktivitas. Jenis-jenis basis data yang umum digunakan antara lain basis data relasional seperti *MySQL* dan *PostgreSQL*, serta basis data *non-relasional* (*NoSQL*) seperti *MongoDB* dan *Firebase Realtime Database*.



Gambar 2.8 Arsitektur Basis Data

(*Source*: https://vycev.wordpress.com/2018/10/05/arsitektur-database/)

Penggunaan basis data, baik dalam bentuk *entity relational database* (ERD) yang telah lama digunakan, *object relational database*, maupun teknologi terbaru seperti *NoSQL* dan *NewSQL*, membutuhkan perancangan dan desain yang tepat. Dengan perencanaan yang matang, basis data dapat diimplementasikan secara optimal dalam sistem informasi dan berfungsi sebagai komponen penting yang menunjang kinerja sistem tersebut. Pemilihan jenis basis data harus disesuaikan dengan kebutuhan aplikasi. Misalnya, basis data relasional cocok digunakan untuk data yang terstruktur dan memiliki relasi kompleks antar tabel, sementara basis data *NoSQL* lebih cocok untuk aplikasi berskala besar dengan kebutuhan data yang fleksibel dan dinamis. Basis data juga dapat dihosting di AWS untuk meningkatkan skalabilitas dan ketersediaan sistem [18].

2.1.9 PostgreSQL

Sistem basis data merupakan metode penyimpanan data yang umum digunakan saat ini. Saat ini, basis data dapat dibagi ke dalam beberapa jenis (misalnya *NoSQL*, berorientasi objek, hierarkis), namun pilihan yang paling umum untuk pengembangan aplikasi adalah basis data relasional. Terdapat banyak sistem manajemen basis data relasional, baik yang gratis maupun komersial, yang tersedia di pasaran dan memiliki perbedaan dalam hal kemampuan serta kecepatan

operasional. Kinerja sistem ini juga dipengaruhi oleh teknologi yang digunakan dalam pembuatan aplikasi yang mengakses basis data, serta sistem operasi tempat aplikasi tersebut dijalankan [19].

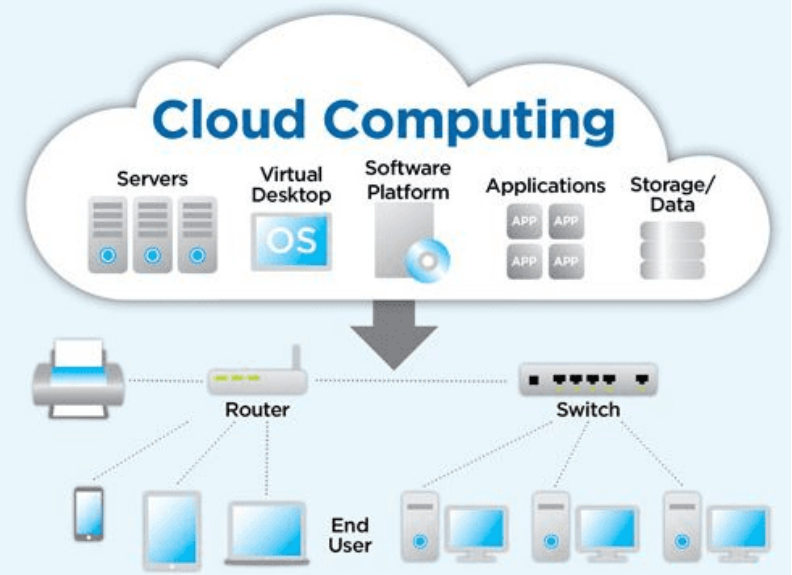
PostgreSQL adalah sistem manajemen basis data relasional (*Relational Database Management System* / RDBMS) yang bersifat *open-source* dan telah dikembangkan selama lebih dari dua dekade. *PostgreSQL* dikenal karena kestabilan, skalabilitas, serta dukungannya terhadap fitur-fitur lanjutan dalam pengelolaan basis data, seperti transaksi ACID (*Atomicity, Consistency, Isolation, Durability*), *indexing*, *constraint* (*foreign key, check, unique*), serta kemampuan menjalankan *query SQL* yang kompleks [20].

Gambar 2.9 Logo *PostgreSQL*
(*Source*: https://id.wikipedia.org/wiki/Berkas:Postgresql_elephant.svg)

Dalam konteks pengembangan backend aplikasi, *PostgreSQL* biasanya digunakan bersama *framework* seperti *Flask* atau *Django* untuk membangun API yang kemudian akan diakses oleh *frontend*. Karena *PostgreSQL* merupakan RDBMS, maka data disimpan dalam bentuk tabel yang saling berelasi. Struktur ini sangat berguna untuk menjaga konsistensi data yang kompleks dan berhubungan erat, seperti transaksi, profil pengguna, atau histori data yang bersifat terstruktur. Jika dibandingkan dengan MySQL dari segi *performance response time*, PostgreSQL lebih unggul dalam mengelola data yang berskala besar dan jumlah record yang banyak [21].

2.1.10 Cloud Computing

Cloud computing merupakan teknologi yang memungkinkan penyimpanan dan pemrosesan data dilakukan melalui internet, tanpa harus memiliki infrastruktur fisik secara langsung. Layanan *cloud* menawarkan berbagai sumber daya komputasi seperti *server*, *database*, jaringan, dan perangkat lunak yang dapat diakses sesuai kebutuhan. Hal ini membuat pengembangan dan pengelolaan aplikasi menjadi lebih fleksibel dan efisien [22].



Gambar 2.10 Arsitektur *Cloud Computing*

(*Source* : https://blog.nashtechglobal.com/know-about-cloud-computing-architecture/)

Keunggulan *cloud computing* terletak pada kemampuannya untuk menskalakan sumber daya secara otomatis sesuai permintaan. Selain itu, teknologi ini juga menawarkan efisiensi biaya karena pengguna hanya membayar berdasarkan penggunaan. Dalam pengembangan sistem modern, *cloud computing* sangat penting untuk memastikan aplikasi dapat berjalan secara optimal dengan dukungan infrastruktur yang handal, aman, dan selalu tersedia.

2.1.11 GitHub

GitHub merupakan platform berbasis *Git* yang digunakan untuk manajemen versi dan kolaborasi dalam pengembangan perangkat lunak. GitHub sejauh ini merupakan platform pengkodean sosial terbesar, yang menampung sejarah pengembangan jutaan *repositori* perangkat lunak kolaboratif, dan menampung lebih dari 73 juta pengguna pada tahun 2021 [23].

Gambar 2.11 Logo Github

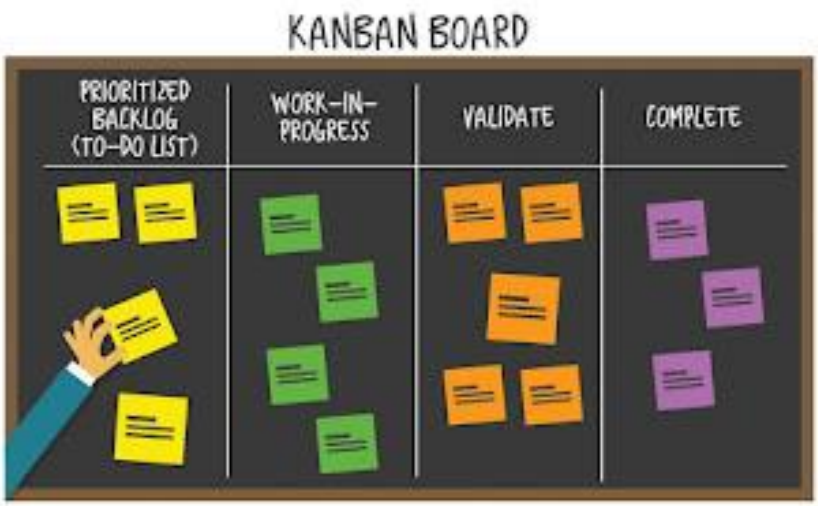
(*Source* : https://www.binar.co.id/blog/apa-itu-github-dan-cara-menggunakannya)

Dalam proyek ini, GitHub digunakan sebagai repositori utama untuk menyimpan seluruh kode sumber *backend*, dokumentasi, serta model *machine learning* (dalam format *JSON*) sebelum *dideploy*. Dengan GitHub, tim pengembang dapat bekerja secara kolaboratif, mengelola versi kode, dan melakukan *pull/push* serta *merge* perubahan dengan aman. Selain itu, fitur issue tracker dan pull request di GitHub membantu dalam mencatat *bug*, mengusulkan perbaikan, serta mengatur *workflow* pengembangan secara lebih terstruktur [24].

2.1.12 Kanban Board

Kanban board adalah metode *visual* untuk manajemen proyek yang digunakan untuk memantau alur kerja (*workflow*) dari setiap tugas atau aktivitas dalam proyek. Dalam proyek skripsi ini, metode Kanban digunakan untuk mengelola dan memantau perkembangan tugas-tugas dalam pengembangan aplikasi, terutama pada bagian *backend*. Papan Kanban biasanya dibagi menjadi beberapa kolom

seperti *Backlog*, *To Do*, *In Progress*, *Testing*, dan *Done*. Setiap tugas atau fitur baru akan dimasukkan ke kolom yang sesuai dan dipindahkan seiring dengan perkembangan proses pengerjaannya. *Kanban Board* dimanfaatkan untuk memetakan alur kerja serta memantau perkembangan proyek dengan menampilkan aktivitas dalam proses pengembangan. Setiap tahap memiliki ruang tersendiri untuk tugas-tugas yang digambarkan melalui kartu-kartu tugas [25].



Gambar 2.12 Struktur *Kanban Board*

(*Source*: http://www.taufanyanuar.com/2020/12/kanban-board.html)

Dengan menggunakan *Kanban board*, tim pengembang dapat melihat secara jelas status dari setiap bagian proyek, memprioritaskan tugas, menghindari *bottleneck*, serta memastikan bahwa seluruh proses pengembangan berlangsung sesuai alur. Penggunaan Kanban juga meningkatkan transparansi dan efisiensi kolaborasi antar anggota tim.

2.1.13 Postman

Postman adalah sebuah platform yang digunakan untuk menguji, mengelola, dan mendokumentasikan *Application Programming Interface* (API). Alat ini menyediakan antarmuka pengguna grafis yang intuitif, sehingga memudahkan pengembang dalam mengirim permintaan HTTP seperti GET, POST, PUT, dan DELETE ke server serta menerima respons secara langsung. Dengan Postman, pengguna dapat mengatur parameter, headers, body, serta autentikasi dalam setiap

request tanpa perlu menulis kode tambahan. Selain itu, Postman juga memungkinkan penyimpanan koleksi *request*, pengelolaan *environment variables*, serta pembuatan skenario pengujian otomatis menggunakan fitur *Collection Runner* dan *Test Scripts*. Hal ini membuat Postman menjadi salah satu alat yang esensial dalam proses pengembangan dan pengujian API berbasis REST [26].

Gambar 2.13 Postman

(*Source*: https://teltics.com/blog/memahami-postman-alat-pengembangan-api-yang-tak-tertandingi)

Dalam konteks pengembangan backend, khususnya pada aplikasi berbasis *machine learning*, Postman sangat membantu untuk memastikan bahwa setiap *endpoint* API berjalan sesuai dengan fungsinya. Misalnya, untuk menguji *endpoint* klasifikasi gambar, pengguna dapat mengirim data gambar ke *backend* dan melihat *respons* hasil klasifikasi yang dikembalikan oleh model *machine learning*. Selain itu, Postman juga mendukung berbagai jenis autentikasi seperti API key dan bearer token, yang sangat berguna dalam pengujian API yang membutuhkan keamanan. Dokumentasi API juga dapat dibuat langsung di Postman dan dibagikan ke tim lain seperti pengembang *frontend* atau tim QA. Dengan berbagai fitur tersebut, Postman tidak hanya berfungsi sebagai alat uji coba, tetapi juga sebagai sarana kolaborasi dalam pengembangan perangkat lunak yang berorientasi pada layanan (*service-oriented*).

2.2 Penelitian Terkait

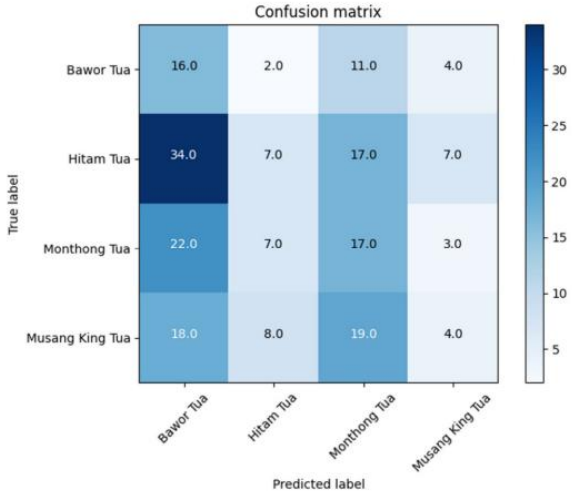
Penelitian ini tidak lepas dari penelitian yang sudah dilakukan, sehingga penelitian yang dilakukan memiliki persamaan dan perbedaan objek yang diteliti. Ringkasan dari penelitian dapat dilihat dibawah ini:

1. M. Rizanda, Mengimplementasikan Convolutional Neural Network (CNN) berbasis arsitektur EfficientNet-B0 untuk melakukan klasifikasi varietas buah durian. Dataset yang digunakan terdiri dari tujuh varietas durian, yaitu *Musang King*, D24/Sultan, *Golden Phoenix*, *Sumatra Super*, Medan, Bengkulu, dan Kota Agung, yang diperoleh dari *Kaggle* serta pengambilan langsung di toko buah durian di Bandar Lampung. Tahapan penelitian meliputi preprocessing citra berupa *resize* menjadi 224×224 piksel, normalisasi, augmentasi data, pembagian dataset, perancangan model, pelatihan, *hyperparameter tuning*, dan evaluasi model. Arsitektur *EfficientNet-B0* dipilih karena memiliki jumlah parameter yang relatif kecil namun tetap mampu memberikan performa klasifikasi yang tinggi dengan memanfaatkan *pretrained weights* dari *ImageNet*. Hasil penelitian menunjukkan bahwa model mampu mencapai akurasi pelatihan sebesar 99,95% dan akurasi validasi sebesar 99,29% setelah dilakukan *hyperparameter tuning*. Penelitian ini membuktikan bahwa *EfficientNet-B0* efektif digunakan untuk klasifikasi varietas durian berbasis citra digital [27].

2. Farhat Febrianto, Mengimplementasikan Convolutional Neural Network menggunakan arsitektur Xception untuk melakukan identifikasi penyakit daun pada tanaman durian. Penelitian ini bertujuan untuk membantu proses deteksi penyakit daun secara otomatis melalui citra digital sehingga dapat mengurangi ketergantungan terhadap identifikasi manual oleh pakar. Dataset yang digunakan berupa citra daun durian yang terdiri dari beberapa kelas penyakit dan daun sehat. Tahapan penelitian meliputi preprocessing citra, perancangan arsitektur *Xception*, pelatihan model dengan pendekatan *transfer learning*, serta evaluasi performa menggunakan *confusion matrix* dan metrik akurasi. Arsitektur Xception dipilih karena memiliki kemampuan ekstraksi fitur yang baik melalui penggunaan *depthwise separable convolution*. Hasil penelitian menunjukkan bahwa model

Xception mampu menghasilkan tingkat akurasi yang tinggi dalam mengklasifikasikan jenis penyakit daun durian [6].

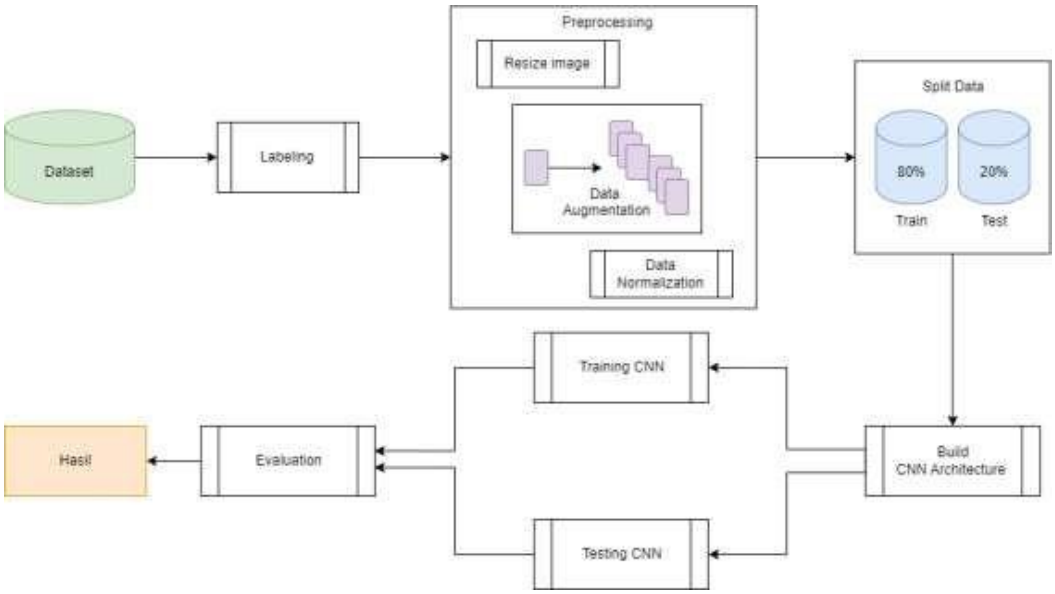
3. Elroy, Sebastian Jody. Nurdiyansyah, Firman. Priyandoko, Gigih, "Klasifikasi Daun Durian Pada Citra Dalam Menentukan Jenis Menggunakan Convolutional Neutral Network" diterbitkan pada tahun 2024. Pada penelitian ini memanfaatkan algoritma *Convolutional Neural Network* (CNN) untuk mengklasifikasikan jenis durian berdasarkan ciri morfologi pada daunnya, dengan hasil menunjukkan adanya peningkatan akurasi seiring bertambahnya jumlah *epoch*. Model yang dibangun berhasil mencapai tingkat akurasi yang cukup tinggi, yaitu sekitar 80% [28].



Gambar 2.14 *Confusion Matrix* Penelitian Terdahulu

4. M.B. Gigih Baskoro Ashari, "Implementasi Algoritma Convolutional Neural Network untuk Meningkatkan Identifikasi Penyakit Tanaman Durian" diterbitkan pada tahun 2024. Dalam penelitian ini berhasil menerapkan algoritma *Convolutional Neural Network* (CNN) dalam upaya meningkatkan proses identifikasi penyakit pada tanaman durian. Berdasarkan hasil pengujian, model CNN mampu mengklasifikasikan lima kategori kondisi daun durian dengan akurasi total sebesar 62%. Namun, kinerja model berbeda-beda pada setiap kelas, di mana presisi tertinggi dicapai pada kategori "Sehat" sebesar 83%, sedangkan presisi terendah terdapat pada kategori "Jamur" sebesar 33%. Temuan ini menunjukkan bahwa model cukup efektif dalam mengenali daun yang sehat, namun masih perlu

dioptimalkan untuk mendeteksi beberapa jenis penyakit seperti jamur dan bercak [16].



Gambar 2.15 Alur Metode Penelitian Peneliti Terdahulu

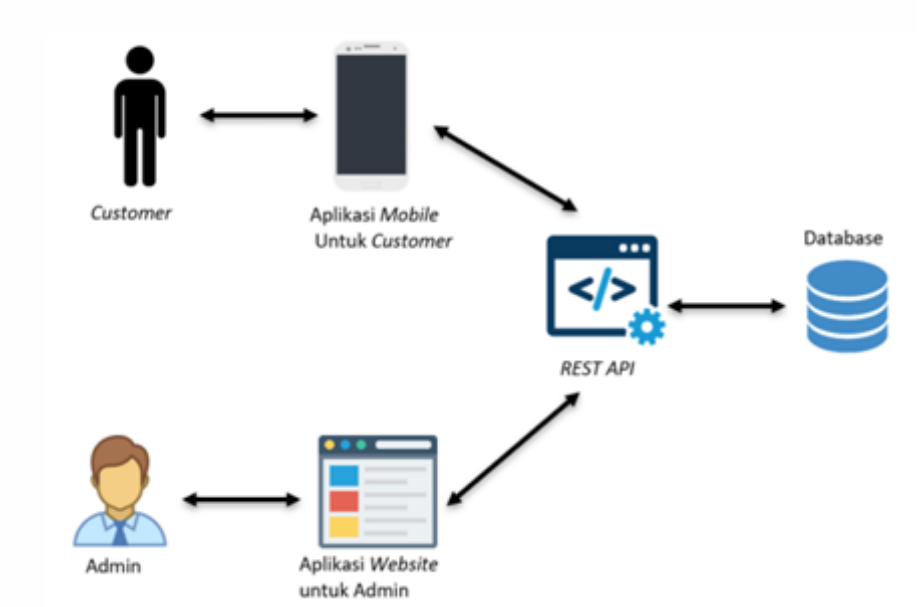
5. Natbais, Yuyun Hana dan Umbu, Ari Bangkit Sanjaya, "Aplikasi Deteksi Penyakit pada Daun Tomat Berbasis Android Menggunakan Model Terlatih Tensorflow Lite" diterbitkan pada tahun 2023. Berdasarkan hasil pengujian dan analisis, dapat disimpulkan bahwa aplikasi pendeteksi penyakit pada daun tomat berbasis Android dengan dukungan model terlatih dari *TensorFlow Lite* telah berhasil dikembangkan. Aplikasi ini mampu mengidentifikasi jenis penyakit daun menggunakan gambar yang diambil melalui kamera perangkat maupun dari galeri. Untuk mendapatkan hasil deteksi yang akurat, kualitas kamera dan sudut pengambilan gambar perlu diperhatikan. Sementara itu, untuk gambar yang diambil dari galeri, penting untuk memastikan bahwa sumber gambar yang digunakan sesuai dan berkualitas [29].

6. G. Nathaniel, "Penerapan Framework Flask sebagai API Dalam Pengembangan Website Prediksi Kebakaran Hutan dan Lahan di Indonesia" diterbitkan pada tahun 2024. Pada penelitian ini Pengujian sistem API dilakukan menggunakan metode *Stress Testing*, yaitu suatu jenis pengujian performa yang bertujuan untuk mengukur kemampuan sistem dalam menangani beban kerja yang melebihi kapasitas normalnya. Pengujian ini bertujuan untuk menilai seberapa tangguh dan stabil

sebuah *website* atau API saat berada dalam kondisi ekstrem. Tujuan utamanya adalah untuk mengetahui batas maksimal sistem dalam merespons permintaan (*request*). Dalam proses ini, parameter yang diuji meliputi jumlah request serta banyaknya virtual user yang digunakan [30].

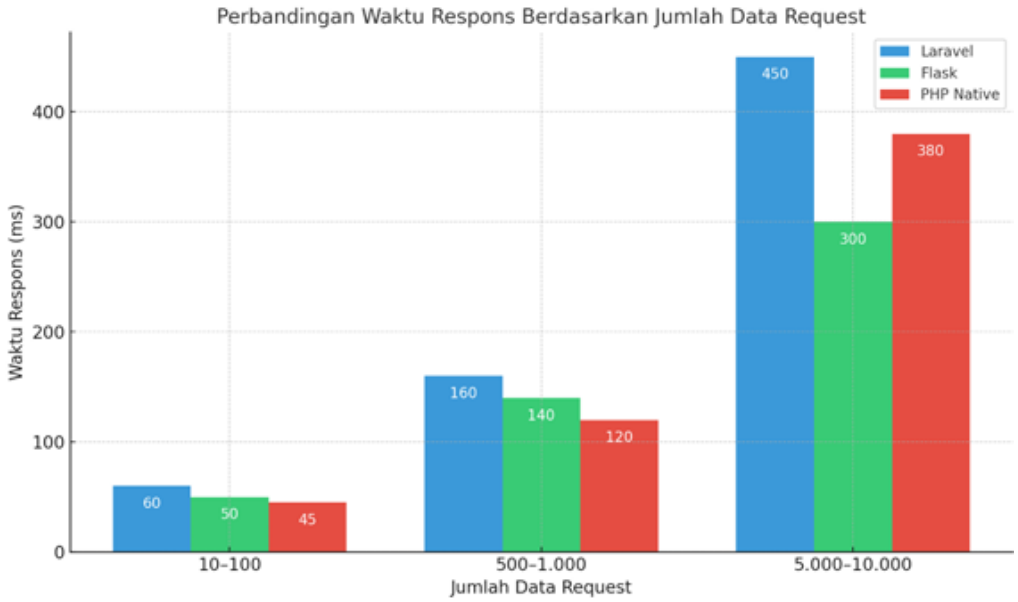
Gambar 2.16 Grafik Hasil dari *Stress Testing* Penelitian Terdahulu

7. I.Arianta, I.Arwani, W.H.Nugraha. "Penerapan REST API dalam Pengembangan Aplikasi Pemesanan Rental Mobil berbasis Web dan Mobile (Studi Kasus: CV. Dwi Cipta Rent Car)" diterbitkan pada tahun 2020. Pada penelitian ini berhasil mengembangkan aplikasi pemesanan rental mobil yang mengatasi inefisiensi operasional CV. Dwi Cipta Rent Car melalui otomatisasi proses transaksi menggunakan REST API untuk mengintegrasikan platform *web* dan *mobile*. Dengan pendekatan *Waterfall*, penelitian ini menghasilkan sistem yang fungsional, kompatibel di berbagai browser dan level API Android, serta didokumentasikan dengan baik melalui diagram UML. Namun, keterbatasan seperti fokus pada satu studi kasus, kurangnya pengujian penerimaan pengguna, dan dukungan hanya untuk Android membatasi generalisasi dan jangkauan aplikasi. Penelitian ini memberikan kontribusi berharga bagi pengembangan sistem rental kendaraan berbasis teknologi, dengan saran untuk pengembangan fitur pelacakan kendaraan dan perluasan platform, yang dapat menjadi dasar untuk penelitian lanjutan dalam meningkatkan efisiensi operasional bisnis sewa kendaraan [31].



Gambar 2.17 Perancangan Arsitektur Sistem Penelitian Terkait

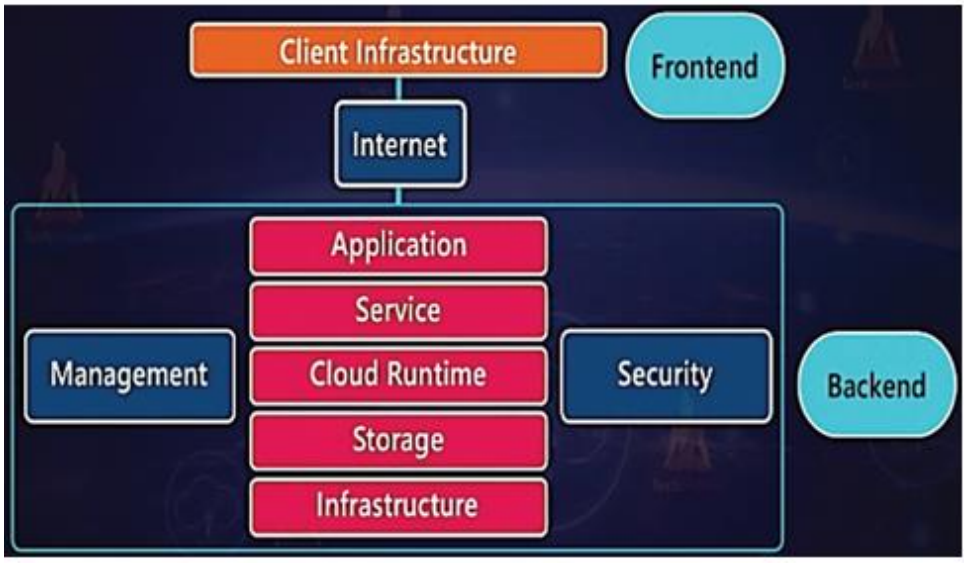
8. S. Andriansyah dan Nurhasanah, "Perbandingan Performa Framework Laravel, Flask API Python, dan PHP Native untuk Aplikasi API pada Data AIS Polbeng" diterbitkan pada tahun 2020. Penelitian ini bertujuan untuk melakukan perbandingan kinerja API yang dikembangkan menggunakan tiga teknologi berbeda, yaitu *Laravel*, *Flask*, dan *PHP Native*, dengan fokus pada pengukuran waktu *respons* berdasarkan jumlah data yang diminta. Pengujian dilakukan untuk mengevaluasi *response time* dari masing-masing API dalam menangani permintaan data dalam jumlah bervariasi, mulai dari skala kecil (10–100 data) hingga skala besar (1.000–10.000 data). Setiap *framework* diuji dengan jumlah data yang berbeda-beda guna menilai pengaruh skala data terhadap performa. Hasil yang diperoleh menunjukkan bahwa *PHP Native* memberikan waktu *respons* tercepat pada data skala kecil hingga menengah (masing-masing 45 ms dan 120 ms), sedangkan *Flask* tampil paling stabil dan efisien saat menangani data dalam jumlah besar dengan rata-rata waktu respons sebesar 300 ms. Sementara itu, *Laravel* meskipun memiliki fitur yang lebih lengkap menunjukkan performa yang lebih lambat, terutama ketika harus memproses data dalam jumlah besar, dengan waktu respons mencapai 450 ms [32].



Gambar 2.18 Grafik Perbandingan Waktu Respon Penelitian Terdahulu

9. J.Tradanarova, W.Hayuhardika, N.Putra "Pembuatan Sistem Informasi Pariwisata Kabupaten Kotawaringin Barat Kalimantan Tengah berbasis Aplikasi Mobile dengan Menggunakan Integrasi REST API" diterbitkan pada tahun 2023. Peneltian ini berhasil mengembangkan aplikasi KOBAR *Connect* untuk mempromosikan pariwisata pasca-COVID-19 di Kotawaringin Barat melalui integrasi REST API antara sistem *web admin* (*Laravel*) dan aplikasi mobile (*Flutter*). Dengan pendekatan *Waterfall*, penelitian ini menghasilkan sistem dengan 94% validitas fungsional dari pengujian *Black Box*, meskipun fitur navigasi gagal karena keterbatasan API *Google Maps*. Skor SUS sebesar 68,88 menunjukkan tingkat kegunaan yang cukup baik namun memerlukan perbaikan untuk mencapai penerimaan yang lebih tinggi. Kelemahan seperti keterbatasan pada *platform Android*, absennya fitur autentikasi, dan fokus pada satu wilayah membatasi skalabilitas sistem. Penelitian ini memberikan kontribusi signifikan dalam mendukung pemulihan pariwisata melalui teknologi digital, dengan saran untuk menambahkan fitur seperti ulasan pengguna dan pengembangan berbasis *web*, yang dapat menjadi landasan untuk penelitian lanjutan dalam sistem informasi pariwisata [33].

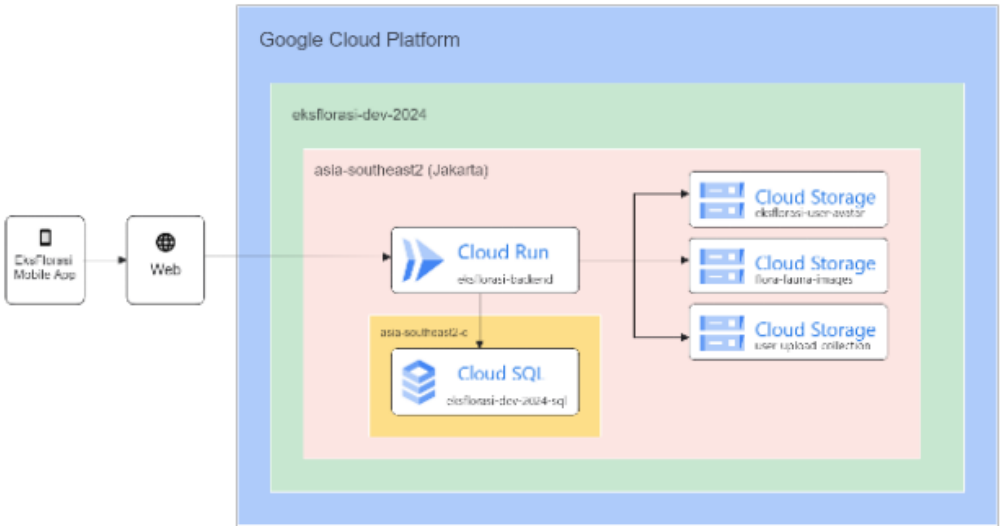
10. D. Patel, “The Role of Amazon Web Services in Modern Cloud Architecture: Key Strategies for Scalable Deployment and Integration” diterbitkan pada tahun 2024. Jurnal ini membahas peran penting *Amazon Web Services* (AWS) dalam membangun arsitektur *cloud modern* yang berskala besar, efisien, dan aman. Penulis menjelaskan bahwa perkembangan *cloud computing* telah merevolusi cara organisasi mengelola infrastruktur TI mereka, dengan menawarkan layanan yang fleksibel, hemat biaya, serta dapat diakses sesuai kebutuhan (*on-demand*). AWS diposisikan sebagai penyedia layanan *cloud* terbesar yang menyediakan berbagai layanan mulai dari komputasi (*compute*), penyimpanan (*storage*), jaringan (*networking*), keamanan (*security*), hingga *machine learning*. Melalui analisisnya, Patel menguraikan beberapa strategi utama untuk mencapai *scalable deployment*. Selain itu, jurnal ini menyoroti pentingnya strategi integrasi AWS, baik dengan sistem *on-premises*, *hybrid cloud*, maupun *multi-cloud* (gabungan beberapa penyedia layanan *cloud*). Pendekatan berbasis API dan data integration dijelaskan sebagai metode untuk memastikan interoperabilitas dan efisiensi data lintas platform [34].



Gambar 2.19. Amazon Web Service Arsitektur Penelitian Terdahulu

11. C.Chandra, F.Wijaya, J.A Gunawan, J.R.Lee, A.Maulana. “Perancangan dan Implementasi RESTful API untuk Aplikasi Mobile Pembelajaran Flora dan Fauna pada *Amazon Web Service*” diterbitkan pada tahun 2024. Penelitian ini berhasil mengembangkan RESTful API sebagai *backend* aplikasi EksFlorasi, yang

mendorong anak-anak usia sekolah untuk belajar tentang flora dan fauna secara interaktif melalui smartphone. Dengan menggunakan *Node.js*, *Express.js*, dan *TensorFlow.js*, API ini mendukung fitur autentikasi, pengumpulan koleksi, dan gamifikasi, sementara deployment pada *Amazon Web Service* (*Cloud Run, Cloud SQL, Cloud Storage*) di region asia-*southeast2* memastikan akses cepat di seluruh Indonesia. Pengujian *black box* dan *load testing* menunjukkan API berfungsi dengan baik hingga 50 pengguna bersamaan dengan tingkat *error* rendah (0,51%), meskipun performa menurun pada beban lebih tinggi. Kelemahan seperti waktu *respons* yang melebihi 1 detik, keterbatasan skalabilitas, dan kurangnya evaluasi pengguna membatasi potensi adopsi. Penelitian ini memberikan kontribusi penting dalam pendidikan berbasis *mobile* dan solusi *cloud*, dengan peluang pengembangan melalui optimisasi performa, dukungan lintas *platform*, dan pengujian kegunaan untuk meningkatkan dampak edukasi dan skalabilitas sistem [35].



Gambar 2.20 Arsitektur *Cloud RESTful API* EksFlorasi Penelitian Terdahulu

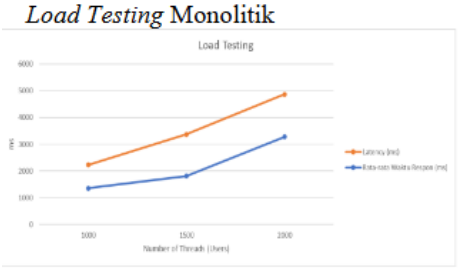
12. Riyanto et al. (2023). Membahas "Analisis Uji Performa Aplikasi Dari Hasil Implementasi Refactoring Arsitektur Monolitik Ke Mikroservis dengan Decomposition dan Strangler Pattern". Dalam arsitektur monolitik, seluruh komponen aplikasi terintegrasi dalam satu kesatuan sehingga menyebabkan penurunan performa ketika jumlah pengguna meningkat serta menyulitkan proses pemeliharaan dan skalabilitas sistem. Untuk mengatasi permasalahan tersebut, penelitian ini menerapkan refactoring arsitektur monolitik menjadi arsitektur

mikroservis menggunakan pendekatan decomposition pattern dan strangler pattern. Decomposition pattern digunakan untuk memecah aplikasi monolitik menjadi beberapa domain bisnis, sedangkan strangler pattern diterapkan secara bertahap melalui tahapan transform, coexist, dan eliminate, sehingga layanan baru dapat menggantikan fungsi lama tanpa menghentikan sistem yang sedang berjalan. Evaluasi kualitas aplikasi dilakukan berdasarkan model ISO/IEC 25010 yang mencakup delapan karakteristik perangkat lunak, seperti performance efficiency, reliability, security, maintainability, dan portability. Pengujian performa dilakukan menggunakan beberapa metode performance testing, yaitu load testing, spike testing, stress testing, dan soak testing dengan bantuan JMeter. Hasil penelitian menunjukkan bahwa aplikasi berbasis mikroservis memiliki waktu respons dan latency yang lebih stabil dibandingkan aplikasi monolitik, terutama pada kondisi jumlah pengguna yang tinggi [36].

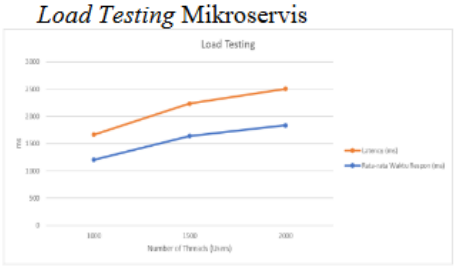
Tabel X. Perbandingan Hasil *Load Testing* Monolitik vs Mikroservis

Number of Threads (Users)	*SmartCampus*(Monolitik)			Mikroservis		
	Status	*Rata-rata Waktu Respon (ms)*	*Latency(ms)*	*Status*	*Rata-rata Waktu Respon (ms)*	*Latency(ms)*
1.000	99,13%	1353,800667	878,904	100%	1204,98	460,096
1.500	99,70%	1813,672	1559,516	100%	1640,249333	593,2733333
2.000	65,30%	3276,332	1587,4905	100%	1837,3275	668,3755

Load Testing Monolitik

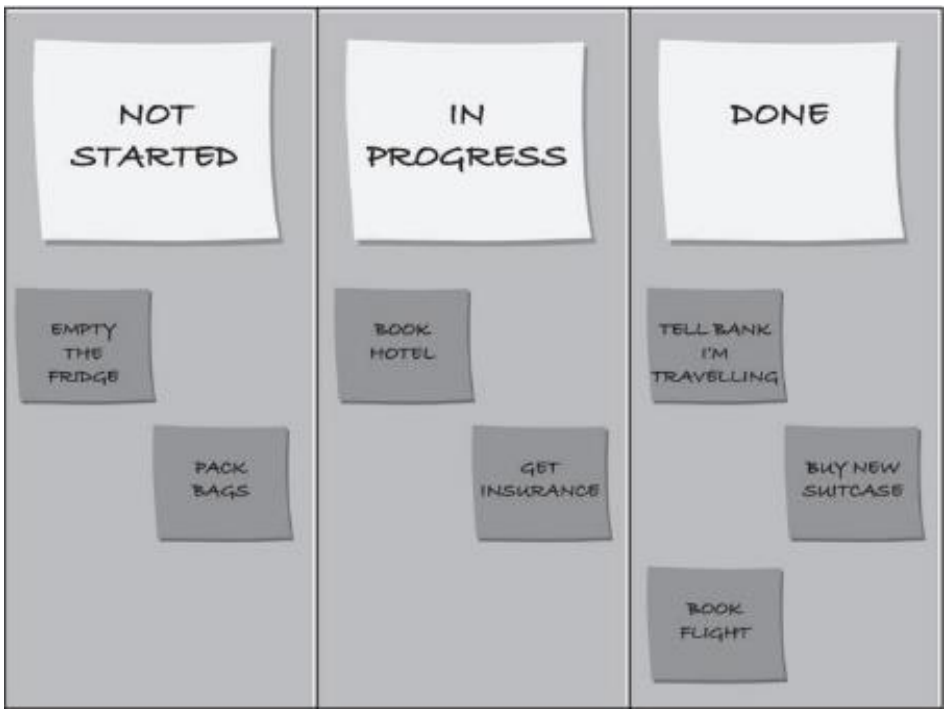


Load Testing Mikroservis



Gambar 2.21. Perbandingan Hasil Load TestingMonolitik vs Mikroservis

13. R.Hartono, “Penerapan Kanban Model Sebagai Metode Perancangan Sistem Informasi (Studi Kasus: Pemetaan Sekolah SMA/K/MA Kota Tasikmalaya)” diterbitkan pada tahun 2022. Pada penelitian ini Metode Kanban merupakan alternatif sederhana namun efektif yang dapat digunakan dalam perancangan sistem informasi. Dengan menggunakan papan Kanban, proses iterasi dapat dibatasi sesuai dengan kebutuhan perancangan yang telah dihitung. Dalam perancangan sistem informasi pemetaan sekolah, proses dilakukan melalui beberapa iterasi yang dibagi ke dalam kolom kanban seperti *task*, *analysis*, *design*, *evaluate*, dan *done*. Setiap iterasi mencakup tahapan seperti inisiasi, perancangan *use case diagram, activity diagram, class diagram,* pemodelan ERD, hingga perancangan antarmuka pengguna. Meskipun awalnya diterapkan pada sistem manufaktur, pendekatan Kanban juga relevan untuk pengembangan sistem informasi [37].



Gambar 2.22 Format Sederhana *Kanban Board*

III.METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Pelaksanaan penelitian dan pengembangan aplikasi ini dilakukan pada waktu dan tempat yang telah ditentukan sebagai berikut:

Waktu : Juni 2025 – Oktober 2025

Tempat : Laboratorium Teknik Komputer Universitas Lampung

Adapun rincian waktu penelitian berdasarkan aktivitas yang dilakukan adalah sebagai berikut:

Tabel 3.1 Waktu dan Tempat Penelitian

No.	Aktivitas	Juni	Juli	Agustus	September	Oktober
1.	Studi Literatur					
2.	Pengumpulan Kebutuhan *(Requirements Gathering)*					
3.	Daftar Pekerjaan *(Backlog)*					
5.	Pekerjaan yang akan dikerjakan *(To Do)*					
6.	Pekerjaan yang sedang dikerjakan *(Work In Progress)*					
7.	Penerapan ke lingkungan produksi *(Deployment Production)*					
8.	Penulisan Laporan					

3.2 Alat Penelitian

Penelitian ini menggunakan beberapa alat yang mendukung pengembangan sistem *backend* untuk aplikasi scan penyakit dan jenis durian. Adapun alat-alat yang digunakan adalah sebagai berikut:

Tabel 3.2. Alat Penelitian

No.	Jenis	Deskripsi	Fungsi
1.	Perangkat Keras	Laptop ASUS TUF Gaming FX506LH (RAM 16GB, Intel Core i5)	Digunakan untuk pengembangan *backend*, pengujian API, dan dokumentasi
2.	*Server Cloud*	*Digital Ocean*	Digunakan untuk *deployment backend* agar dapat diakses oleh aplikasi *Android*
3.	Sistem Operasi	*Windows* 10 / *Ubuntu* 22.04	Lingkungan kerja untuk pengembangan *backend*
4.	Bahasa Pemrograman	*Python* 3.10	Digunakan untuk menulis program *backend*
5.	*Framework*	*Flask*	*Framework* utama untuk membuat REST API *backend*
6.	*Database*	*PostgreSQL*	Menyimpan data pengguna, hasil klasifikasi, dan pengingat

3.3 Struktur Tim

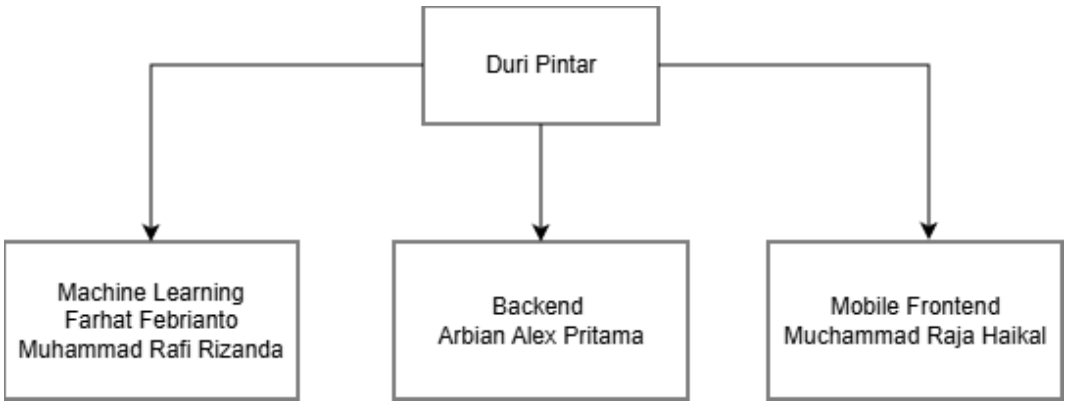
Tim *Artificial Intelligence* (AI) yang terdiri dari Farhat Febrianto dan Muhammad Rafi Rizanda, berperan krusial dalam mengembangkan inti kecerdasan aplikasi. Farhat bertanggung jawab khusus untuk pembuatan model klasifikasi penyakit daun pada tanaman durian. Model ini akan memungkinkan aplikasi mengidentifikasi dan mengklasifikasikan berbagai penyakit yang menyerang daun durian. Sementara itu, Rafi ditugaskan untuk pembuatan model klasifikasi jenis durian. Model ini akan memungkinkan aplikasi mengenali dan membedakan berbagai varietas durian.

Model-model AI ini selesai dikembangkan, peran Arbian Alex Pritama sebagai *Backend Developer* menjadi sangat vital. Arbian bertanggung jawab untuk membuat API (*Application Programming Interface*) yang berfungsi sebagai jembatan komunikasi antara model-model AI yang telah dibuat dan aplikasi *mobile*. API ini akan memungkinkan aplikasi *mobile* untuk mengirim data (misalnya, gambar daun atau durian) ke model AI dan menerima hasil klasifikasi kembali.

Muchammad Raja Haikal Fiaugustian sebagai *Mobile Developer* bertanggung jawab untuk membuat interface aplikasi *mobile*. Raja akan mendesain dan

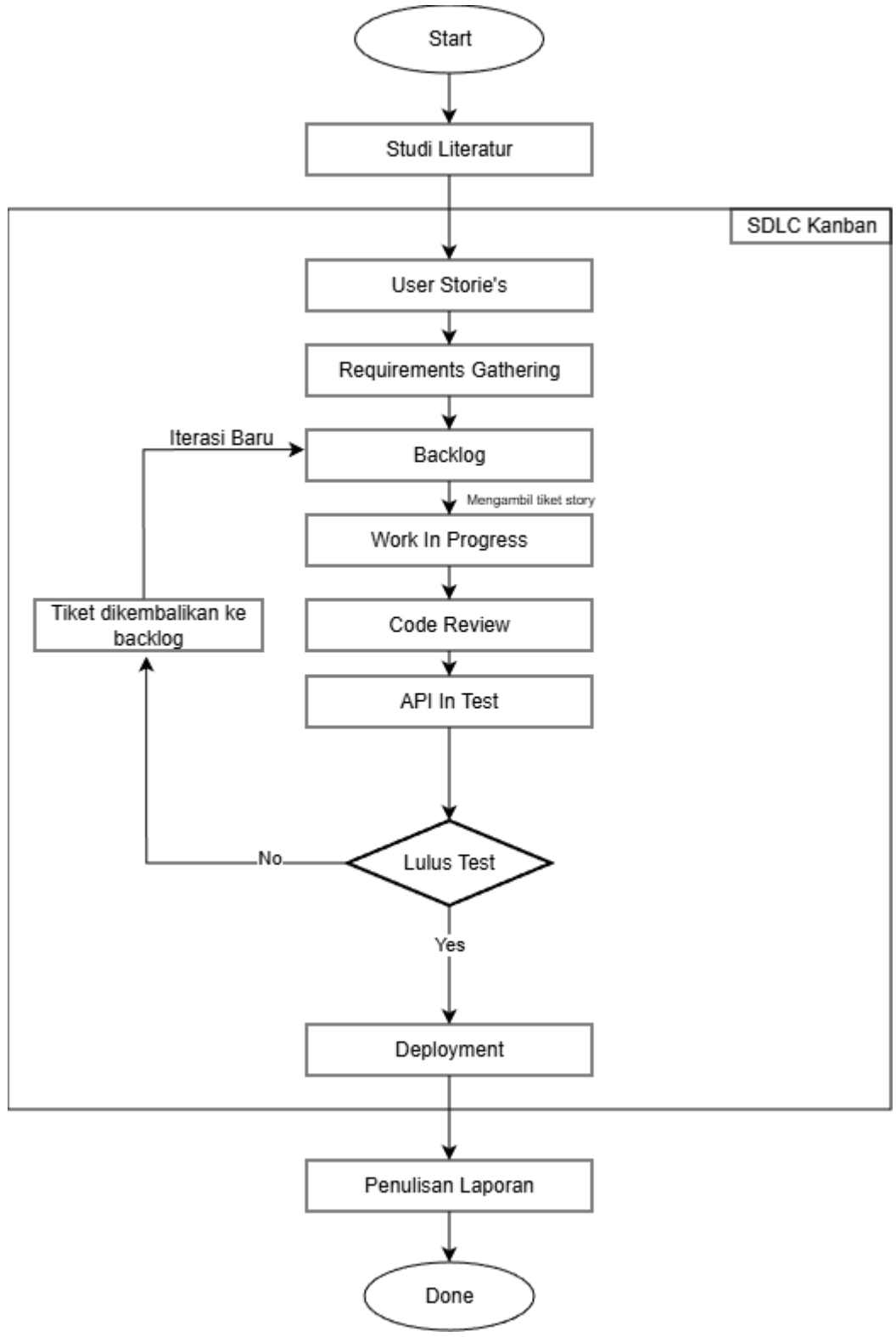
mengembangkan tampilan depan aplikasi yang intuitif dan mudah digunakan oleh pengguna. Ini termasuk merancang antarmuka untuk mengunggah gambar, menampilkan hasil klasifikasi dari model AI melalui API yang disediakan oleh Arbian, dan fitur-fitur lain yang relevan.

Tim AI (Farhat dan Rafi) mengembangkan model-model inti; Arbian (*Backend*) membangun API untuk menghubungkan model-model tersebut; dan Raja (*Mobile*) menciptakan antarmuka pengguna yang berinteraksi dengan API untuk menampilkan hasil dari model AI. Keterkaitan ini memastikan bahwa model AI dapat diakses dan dimanfaatkan secara efektif oleh pengguna melalui aplikasi *mobile*, menciptakan sistem yang terintegrasi dan fungsional. Adapun struktur tim dalam penelitian ini adalah sebagai berikut:



Gambar 3.1 Struktur Tim

3.4 Tahapan Penelitian



Gambar 3.2 Tahapan Penelitian

Berdasarkan Gambar 3.2 Tahapan Penelitian ini menggunakan pendekatan *Software Development Life Cycle* (SDLC) dengan model pengembangan Kanban. Metode Kanban dipilih karena bersifat fleksibel dan memungkinkan proses pengembangan dilakukan secara bertahap serta berulang (iteratif) sesuai dengan kebutuhan sistem. Alur tahapan penelitian yang dilakukan dapat dijelaskan sebagai berikut.

3.4.1 Studi Literatur

Tahap studi literatur dalam penelitian ini dilakukan dengan menelaah berbagai sumber referensi yang relevan dengan topik pengembangan sistem berbasis *machine learning* dan *backend* API menggunakan *framework* Flask. Referensi yang digunakan berasal dari jurnal ilmiah nasional maupun internasional, buku teks terkait kecerdasan buatan, serta dokumentasi resmi seperti Flask, TensorFlow, dan OpenWeather API.

Kegiatan studi literatur ini bertujuan untuk memperdalam pemahaman penulis terhadap beberapa aspek utama, yaitu:

1. Penerapan arsitektur model *Xception* yang digunakan untuk klasifikasi gambar daun durian.
2. Implementasi *framework* Flask sebagai *framework* untuk pembuatan *backend* API yang efisien dan ringan.
3. Integrasi API eksternal (OpenWeather API) untuk menampilkan informasi cuaca secara *real time*.
4. Penerapan metode Kanban sebagai pendekatan manajemen proyek dalam pengembangan perangkat lunak, yang digunakan untuk membantu perencanaan, pembagian tugas, dan pemantauan progres pengembangan sistem.

Selain memperkuat landasan teori, tahap ini juga digunakan untuk mengidentifikasi kebutuhan sistem dan aktor yang terlibat dalam pengembangan aplikasi. Berdasarkan hasil kajian literatur dan analisis kebutuhan, aktor utama yang berinteraksi dengan sistem terdiri dari:

1. Admin atau *Developer*, yang bertugas melakukan pengelolaan sistem, memperbarui model *machine learning*, dan melakukan *deployment* ke *server*.

2. Pengguna (*User*), yaitu petani atau masyarakat umum yang menggunakan aplikasi untuk mendeteksi penyakit dan jenis durian melalui gambar daun.

3.4.2 Metode Kanban

Kanban merupakan metode yang berfokus pada visualisasi proses dan pembatasan jumlah pekerjaan yang sedang berlangsung (*Work in Progress*/WIP), sehingga dapat meningkatkan efisiensi dan mengurangi hambatan dalam proses pengembangan. Metode Kanban dibagi ke dalam beberapa tahapan utama yang divisualisasikan dalam papan Kanban menggunakan aplikasi *Notion*, yaitu :

a. Backlog : Berisi semua *user story* yang berasal dari hasil tahapan *requirement gathering*, seperti fitur *login*, klasifikasi gambar daun dan buah, penyimpanan hasil *scan*, serta pengingat. Pada tahap ini, semua tugas dikumpulkan tanpa memperhatikan prioritas pengerjaan terlebih dahulu.
b. To Do : Merupakan daftar tugas yang telah diprioritaskan untuk dikerjakan pada waktu tertentu. Tugas yang berpindah ke tahap ini adalah telah memiliki desain awal atau draft API *contract*, serta siap dikembangkan.
c. Work In Progress : Tahap ini mencakup tugas yang sedang dalam proses pengerjaan. Karena hanya satu orang yang mengembangkan *backend*, maka hanya satu hingga dua subtask dikerjakan sekaligus untuk menjaga efisiensi.
d. Done : Setelah *subtask* selesai, diuji menggunakan *Postman*, dan lulus *code review*, maka tugas tersebut dipindahkan ke kolom *Done*.

3.4.2.1 User Stories

Tahapan *user story* dilakukan setelah tahap studi literatur, dengan tujuan untuk mendefinisikan kebutuhan pengguna secara lebih spesifik melalui sudut pandang pengguna akhir. Pada tahap ini, peneliti berfokus untuk memahami perilaku, kebutuhan, serta harapan pengguna terhadap aplikasi yang dikembangkan, yaitu petani durian yang ingin memanfaatkan teknologi untuk membantu proses pemantauan kondisi tanaman.

Proses perumusan user story dilakukan berdasarkan hasil analisis kebutuhan pengguna, studi lapangan ringan dengan petani durian di wilayah Bumi Kedaton, serta diskusi internal dengan tim pengembang. Setiap *user story* dirumuskan menggunakan pendekatan naratif yang menjelaskan tiga komponen utama, yaitu: siapa pengguna (aktor), apa yang dibutuhkan (fitur), dan mengapa kebutuhan tersebut penting (tujuan penggunaan).

Enam user story yang disusun merepresentasikan kebutuhan inti dari sistem Duri Pintar, meliputi proses autentikasi pengguna, pemindaian gambar daun atau buah durian, klasifikasi hasil deteksi penyakit dan jenis durian, penyimpanan riwayat hasil pemindaian, serta akses terhadap informasi cuaca terkini melalui integrasi dengan *OpenWeather* API.

Hasil dari tahap *user story* ini kemudian dijadikan dasar untuk proses *requirement gathering* dan disusun ke dalam *product backlog* pada papan Kanban Notion. Dengan demikian, setiap user story dapat direalisasikan secara bertahap dan terukur selama proses pengembangan sistem *backend* API Duri Pintar.

Tabel 3.3 User Story

No.	ID	User Story
1.	US-01	Sebagai pengguna, saya ingin dapat *login* agar saya bisa mengakses fitur aplikasi secara personal.
2.	US-02	Sebagai pengguna, saya ingin dapat memindai gambar daun durian agar saya bisa mengetahui apakah tanaman saya terkena penyakit.
3.	US-03	Sebagai pengguna, saya ingin dapat memindai gambar buah durian agar saya bisa mengetahui jenis durian yang saya tanam.
4.	US-04	Sebagai pengguna, saya ingin melihat hasil klasifikasi secara langsung setelah memindai gambar.
5.	US-05	Sebagai pengguna, saya ingin menyimpan dan melihat riwayat scan sebelumnya agar dapat melacak kondisi tanaman dari waktu ke waktu.
6.	US-06	Sebagai pengguna, saya ingin melihat informasi cuaca terkini yang berkaitan dengan lokasi saya agar saya bisa menentukan waktu yang tepat untuk perawatan tanaman.

3.4.2.2 Requirements Gathering

Tahap *requirements gathering* dilakukan untuk mengidentifikasi seluruh kebutuhan sistem *backend* yang akan dibangun, berdasarkan alur dan fitur aplikasi Durian App. Tahap *Requirements Gathering* memiliki peran krusial karena *output*-nya berupa tujuan atau target yang harus dicapai oleh aplikasi, yang dirumuskan berdasarkan *user stories*. Proses ini mencakup beberapa bagian, antara lain identifikasi kebutuhan fungsional, kebutuhan non-fungsional, perancangan *use case diagram*, perancangan basis data, desain arsitektur sistem, pemetaan sub-tugas, serta penyusunan API *contract*.

a) Analisis Kebutuhan Pengguna

Dilakukan diskusi bersama *stakeholder* membahas temuan – temuan dari hasil studi literatur untuk kebutuhan petani.

Terdapat dua kelompok stakeholder utama yang berperan dalam proses pengembangan sistem:

1. Developer (Mahasiswa Peneliti): Berperan sebagai pengembang utama yang melakukan perancangan, pembuatan kode *backend*, integrasi model *machine learning*, pengujian API, serta *deployment* ke *server*. Dalam metode Kanban, developer mengatur dan memindahkan tugas melalui kolom *To Do*, *In Progress*, dan *Complete* di Notion untuk memantau progres pekerjaan.
2. Dosen Pembimbing / Reviewer Teknis: Berperan sebagai pengawas dan pemberi umpan balik pada setiap iterasi pengembangan sistem. Dalam metode Kanban, dosen pembimbing berfungsi mengevaluasi hasil kerja yang sudah selesai dan memberikan persetujuan sebelum tahap berikutnya dimulai.

Kedua *stakeholder* ini berperan penting untuk memastikan bahwa alur kerja berjalan efisien, tidak terjadi penumpukan pekerjaan, dan hasil pengembangan sesuai dengan tujuan penelitian. Kemudian dari hasil diskusi, diperoleh daftar kebutuhan pengguna yang akan diterapkan dalam perancangan sistem ini. Selanjutnya, kebutuhan pengguna tersebut diuraikan lebih lanjut menjadi kebutuhan fungsional dan non-fungsional. Adapun daftar kebutuhan pengguna hasil dari diskusi tim perancangan dengan petani adalah sebagai berikut:

a. Dapat mengklasifikasikan jenis durian berdasarkan gambar buah yang dipindai.
b. Dapat mendeteksi penyakit daun durian melalui pemindaian daun.
c. Dapat melihat informasi cuaca terkini untuk membantu dalam perawatan pohon durian.
d. Dapat melihat harga durian di pasaran sebagai acuan penjualan.
e. Dapat menyimpan dan melihat riwayat hasil *scan* untuk memantau kondisi tanaman.

b) Kebutuhan Fungsional

Tabel 3.4 Kebutuhan Fungsional

ID	Penjelasan
KF-01	Sistem dapat melakukan proses *login* dan *logout* untuk pengguna terdaftar.
KF-02	Sistem dapat melakukan pemindaian gambar buah durian
KF-03	Sistem dapat melakukan pemindaian gambar daun durian
KF-04	Sistem dapat menampilkan hasil klasifikasi jenis durian kepada pengguna.
KF-05	Sistem dapat menampilkan jenis penyakit daun yang terdeteksi kepada pengguna.
KF-06	Sistem dapat memungkinkan pengguna untuk melihat riwayat hasil deteksi

c) Kebutuhan Non-Fungsional

Tabel 3.5 Kebutuhan Non-Fungsional

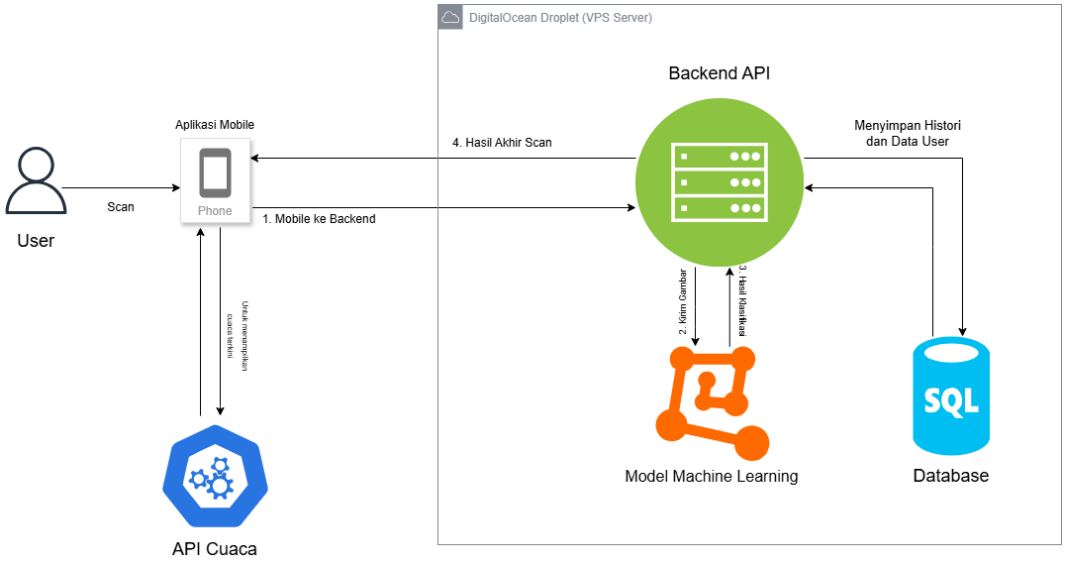
ID	Parameter	Penjelasan
KnF-01	*Availability*	Sistem dapat tersedia dan dapat diakses oleh pengguna selama 24 jam dalam sehari tanpa gangguan
KnF-02	*Reliability*	1. Sistem dapat memberikan hasil klasifikasi durian dan deteksi penyakit daun secara konsisten. 2. Sistem dapat menyimpan dan menampilkan riwayat hasil *scan* pengguna.
KnF-03	*Performance*	1. Sistem *backend* mampu memberikan respons API dalam waktu yang cepat. 2. Sistem mampu menangani beberapa permintaan pengguna secara bersamaan tanpa mengalami penurunan kinerja yang signifikan.
KnF-04	*Portability*	Sistem *backend* dapat dijalankan pada berbagai lingkungan sistem operasi seperti *mac* atau *Windows* selama telah tersedia dependensi yang dibutuhkan, seperti Python, Flask, TensorFlow, dan PostgreSQL.
KnF-05	*Safety*	1. Sistem menerapkan autentikasi pengguna menggunakan *JSON Web Token* (JWT). 2. *Password* pengguna disimpan dalam bentuk *hash* untuk menjaga keamanan data. 3. Akses ke basis data dibatasi menggunakan akun dengan hak akses tertentu.
KnF-06	*Usability*	1. Sistem menyediakan *endpoint* REST API dengan struktur yang konsisten dan mudah dipahami. 2. Sistem menyediakan dokumentasi API melalui *Postman Collection* untuk memudahkan proses integrasi dan pengembangan aplikasi *client*.

d) Arsitektur Sistem

Arsitektur sistem aplikasi Durian Pintar dirancang menggunakan pendekatan *client–server* yang diimplementasikan pada lingkungan VPS *DigitalOcean*. Sistem ini terdiri dari beberapa komponen utama, yaitu aplikasi *mobile*, *backend* API,

model *machine learning*, API cuaca, dan basis data. Aplikasi *mobile* berfungsi sebagai antarmuka pengguna untuk melakukan pemindaian daun durian serta menampilkan hasil klasifikasi penyakit, varietas durian, dan informasi cuaca terkini. Ketika pengguna melakukan proses pemindaian, data gambar dikirimkan ke *backend* API yang dibangun menggunakan *framework* Flask. *Backend* kemudian meneruskan data tersebut ke model *machine learning* untuk melakukan prediksi, baik deteksi penyakit maupun klasifikasi jenis durian. Hasil prediksi dikembalikan ke *backend* untuk disimpan ke dalam basis data bersama dengan riwayat pemindaian pengguna. Selain itu, *backend* juga terhubung dengan API cuaca eksternal untuk memperoleh data kondisi cuaca yang akan ditampilkan pada aplikasi *mobile*. Semua data pengguna, hasil prediksi, serta informasi cuaca tersimpan dalam basis data SQL yang di-*host* di *server* yang sama dengan *backend* API. Dengan rancangan arsitektur ini, sistem dapat beroperasi secara efisien dan terintegrasi, di mana aplikasi *mobile* hanya berperan sebagai *klien* ringan sementara seluruh proses komputasi utama dilakukan di sisi *server*.

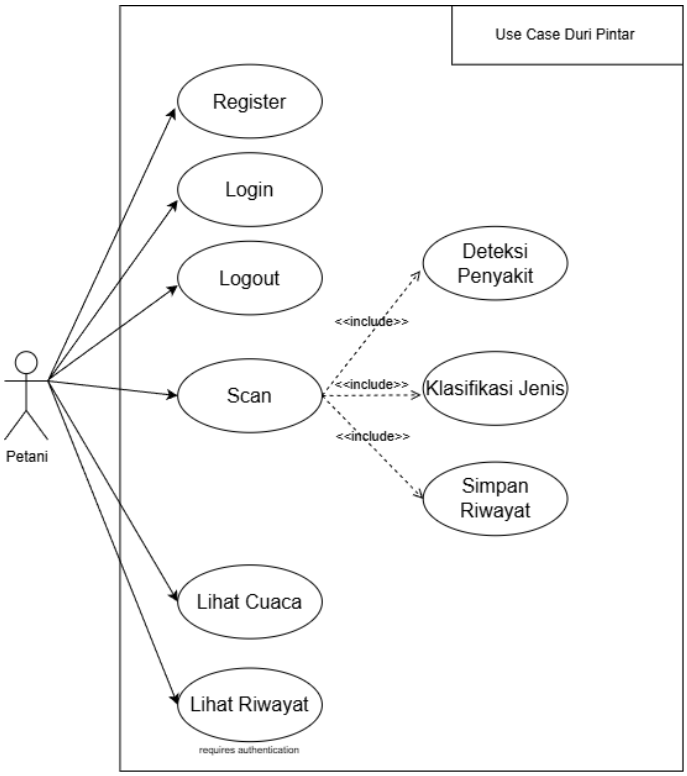
Arsitektur sistem *backend* dalam Durian Apps adalah sebagai berikut:



Gambar 3.3 Arsitektur Diagram

e) Use Case Diagram

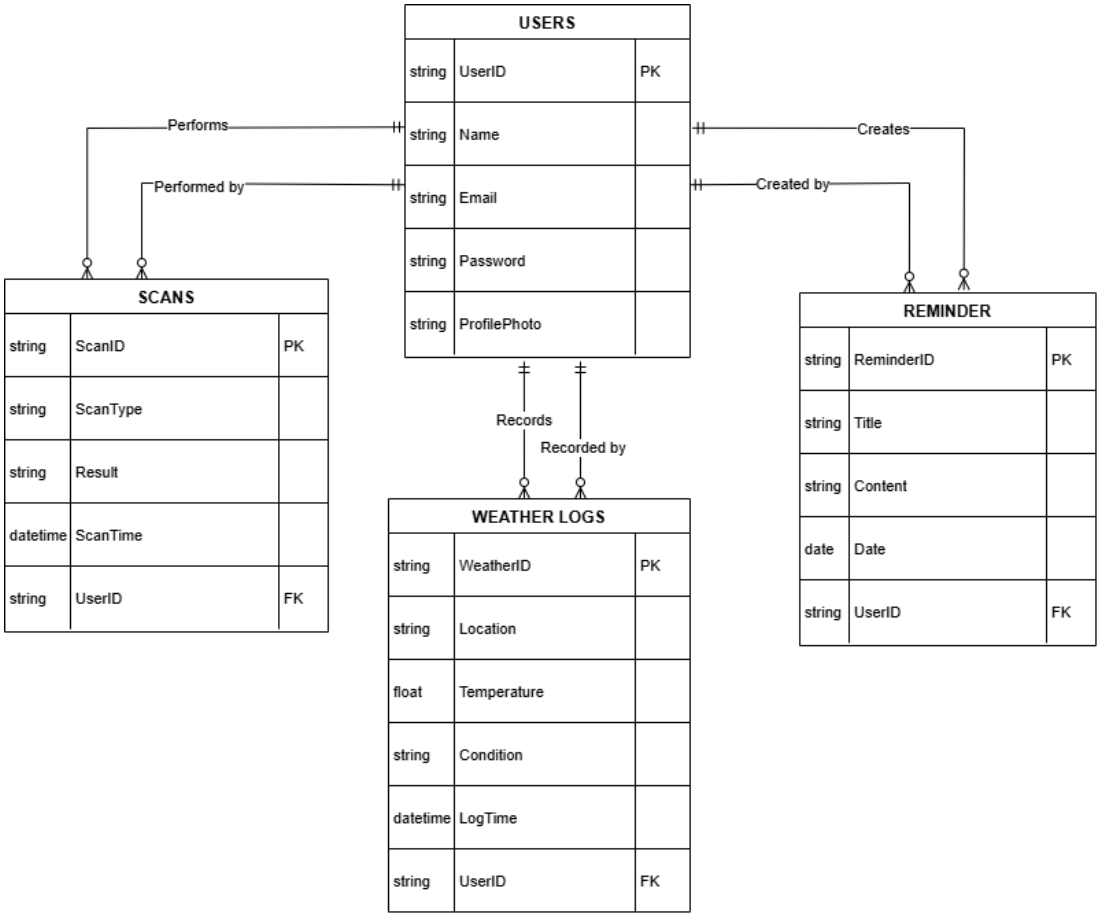
Dalam merancang *use case diagram*, penting untuk menggambarkan hubungan antara fungsi-fungsi yang disediakan oleh sistem dan para aktor yang berinteraksi dengannya. *Use case diagram* berfungsi sebagai alat visualisasi untuk menunjukkan bagaimana pengguna (aktor) memanfaatkan kapabilitas sistem. Selain itu, diagram ini juga membantu dalam mengidentifikasi dan memahami keterkaitan antar *use case* dalam sistem. Berikut merupakan gambaran *use case diagram* untuk sistem yang akan dikembangkan.



Gambar 3.4 Usecase Diagram

f) Entity Relationship Diagram

Pada tahap perancangan basis data, akan ditentukan tabel-tabel yang diperlukan oleh sistem beserta kolom dan tipe data yang sesuai. Desain *database* ini juga akan menampilkan hubungan antar tabel, termasuk relasi satu-ke-satu (*one-to-one*), satu-ke-banyak (*one-to-many*), hingga banyak-ke-banyak (*many-to-many*). Gambaran hubungan tersebut disajikan dalam bentuk *entity relationship diagram* (ERD).



Gambar 3.5 Entity Relationship Diagram

g) Subtask

Setelah melakukan identifikasi kebutuhan fungsional, langkah selanjutnya adalah melakukan pemetaan *subtask*. *Subtask* merupakan rincian pekerjaan teknis yang dibutuhkan untuk merealisasikan setiap fitur dalam sistem. Pemecahan ini dilakukan agar proses pengembangan dapat berjalan lebih terstruktur, terukur, dan efisien sesuai dengan prinsip Kanban, di mana setiap *subtask* dapat dikelola secara modular. Tabel berikut menyajikan daftar subtask berdasarkan kebutuhan fungsional:

Tabel 3.6 *Subtask*

ID	Subtask	Deskripsi
ST-01	*Setup Environment*	Mengatur virtual *environment*, menginstal *Flask*, dan dependensi lainnya seperti *TensorFlow* dan *psycopg2*
ST-02	Struktur *Folder*	Menyusun Direktori *Backend*
ST-03	Koneksi *Database*	Membuat koneksi *Flask* ke PostgreSQL menggunakan SQLAlchemy atau *psycopg2*
ST-04	*Endpoint login/logout*	Membuat API untuk autentikasi pengguna
ST-05	*Update Profile*	Membuat API untuk mengubah nama atau foto pengguna
ST-06	*Scan* Daun	Menerima input gambar daun dan melakukan klasifikasi penyakit
ST-07	*Scan* Buah	Menerima input gambar buah dan melakukan identifikasi jenis durian
ST-08	Simpan Hasil	Menyimpan hasil klasifikasi
ST-09	Lihat Riwayat	Mengambil riwayat hasil *scan* pengguna dari *database*
ST-10	Tambah *Reminder*	Membuat API untuk menyimpan pengingat perawatan pohon
ST-11	Lihat *Reminder*	Mengambil daftar *reminder* milik pengguna
ST-12	*Deployment*	Men-*deploy backend* ke *Digital Ocean* dan melakukan pengujian *endpoint* produksi

h) API *Contract*

API *Contract* adalah dokumen teknis yang menjelaskan struktur komunikasi antara *klien* (*mobile app*) dan *server* (*backend Flask*). Setiap *endpoint* dijelaskan dari segi metode HTTP, *format request*, dan struktur *response*. Dokumentasi ini penting agar proses integrasi antara *frontend* dan *backend* dapat berjalan konsisten serta meminimalkan miskomunikasi antar pengembang.

Tabel 3.7 API Contract

No	Fungsi	Method	Endpoint
1.	Autentikasi pengguna dan menghasilkan *token* akses	POST	/api/login
2.	Mengubah data pengguna seperti nama dan foto	PUT	/api/profile/update
3.	Melakukan prediksi penyakit daun durian berdasarkan gambar	POST	*/api/scan/daun*
4.	Mengklasifikasikan jenis buah durian berdasarkan gambar	POST	*/api/scan/buah*
5.	Mengambil seluruh riwayat *scan* (buah dan daun) milik pengguna	GET	*/api/scan/history*
6.	Menyimpan pengingat untuk perawatan pohon durian	POST	*/api/reminder*
7.	Mengambil seluruh pengingat milik pengguna	GET	*/api/reminder/list*

3.4.2.3 Backlog

Backlog merupakan daftar yang menampung seluruh tugas atau pekerjaan yang akan dikerjakan dalam proyek. Anggota tim dapat memilih tugas dari *backlog* untuk kemudian dikerjakan. Sifat *backlog* bersifat fleksibel atau dinamis, artinya daftar ini dapat terus diperbarui seiring berjalannya proyek, misalnya karena munculnya kebutuhan baru atau adanya masukan dari pihak-pihak terkait. Isi dari *backlog* umumnya berupa *user stories* yang telah dikumpulkan pada tahap *requirement gathering* dan berfungsi sebagai panduan dalam proses implementasi. Selain itu, *backlog* juga dapat mencakup laporan *bug* dari lingkungan produksi, selama *bug* tersebut tidak bersifat kritis atau menghambat jalannya pengembangan.

3.4.2.4 Work In Progress

Setelah *backlog* disusun, proses pengembangan perangkat lunak dapat memasuki tahap implementasi. Karena pengembangan aplikasi ini menerapkan metode Kanban dalam siklusnya, maka jumlah tugas yang dapat dikerjakan secara bersamaan perlu dibatasi. Dalam konteks ini, batasan tersebut ditentukan oleh jumlah anggota tim *backend*. Pengerjaan/pembuatan API untuk aplikasi *mobile* ini

menggunakan bahasa pemrograman *flask*. Karena hanya ada satu orang yang menangani bagian *backend*, maka hanya satu tugas yang dapat dikerjakan pada satu waktu. Pada tahap ini, pengembang akan menerima satu *user story* beserta detail desain yang telah ditetapkan untuk kemudian diubah menjadi kode program.

3.4.2.5 Code Review

Pada tahap *code review*, kode yang telah dikembangkan akan ditinjau kembali guna memastikan bahwa penulisannya sudah sesuai dan bebas dari kesalahan yang dapat menyebabkan kegagalan saat dijalankan. Selain itu, tahap ini juga dimanfaatkan untuk mengevaluasi apakah fitur yang dikembangkan berfungsi sebagaimana mestinya. Sebagai bukti pengerjaan, pengembang dapat menyertakan tangkapan layar atau dokumentasi serupa. Proses ini dilakukan menggunakan alat bernama *Git* dengan membuat *pull request*. *Pull request* inilah yang akan diperiksa dan divalidasi guna menjaga kualitas dan konsistensi kode dalam proyek.

3.4.2.6 Software In Test

Pada tahap *software in test*, seluruh tugas yang telah diselesaikan dan melewati proses *code review* akan diuji menggunakan alat bantu *Postman*. *Postman* digunakan untuk menguji setiap *endpoint* REST API yang telah dikembangkan, dengan cara mengirimkan berbagai jenis permintaan HTTP dan memverifikasi apakah *respons* yang diterima telah sesuai dengan spesifikasi yang diharapkan. Pengujian ini mencakup berbagai skenario seperti pengiriman data yang valid, data tidak valid, autentikasi gagal, serta parameter yang tidak lengkap. Tujuan utama dari tahap ini adalah untuk memastikan bahwa API dapat berjalan dengan benar dan stabil sesuai kebutuhan fungsional. Jika terdapat ketidaksesuaian hasil atau kesalahan selama pengujian, maka tugas akan dikembalikan ke backlog untuk diperbaiki. Namun, apabila semua pengujian berjalan dengan baik dan hasilnya sesuai, maka tugas dapat dilanjutkan ke tahap deployment staging.

3.4.2.7 Deployment Production

Tahap *deployment production* merupakan langkah akhir dalam pengembangan sistem, di mana *backend* aplikasi yang telah melewati proses pengujian di-*deploy* ke lingkungan produksi agar dapat diakses oleh pengguna secara publik melalui aplikasi *mobile*. Pada penelitian ini, *deployment* dilakukan ke layanan *Digital Ocean*, yaitu platform *cloud* berbasis *Infrastructure as a Service* (IaaS) dari *Digital Ocean* yang memungkinkan penggunaan *virtual machine* (VM) sebagai *server backend*.

Instansi *virtual machine* dikonfigurasi dengan sistem operasi Ubuntu 22.04 LTS, kemudian *backend Flask* dijalankan menggunakan *Gunicorn* sebagai *web server gateway interface* (WSGI). *Nginx* digunakan sebagai *reverse proxy* yang mengatur lalu lintas permintaan ke aplikasi *Flask* serta memastikan akses ke API tersedia melalui port 80. Setelah proses *deployment* selesai, API *backend* dapat diakses melalui IP publik atau domain khusus, sehingga dapat dihubungkan langsung dengan aplikasi Android.

Model *machine learning Xception* yang telah dilatih sebelumnya di-*load* ke *backend* saat *server* dijalankan, dan siap menerima input gambar dari pengguna untuk diklasifikasikan. Hasil klasifikasi disimpan ke basis data *PostgreSQL* yang terintegrasi di dalam VM yang sama atau dalam instansi database terpisah yang terhubung langsung. Konfigurasi keamanan seperti *firewall* dan otorisasi akses juga diatur agar hanya permintaan dari aplikasi *mobile* yang valid yang dapat diterima oleh API. Pengujian lanjutan dilakukan pasca-*deployment* untuk memastikan bahwa semua *endpoint* berfungsi sesuai ekspektasi, mulai dari *login*, klasifikasi gambar, hingga penyimpanan histori hasil *scan* ke dalam *database*.

3.4.3 Penulisan Laporan

Langkah akhir dalam penelitian ini adalah penyusunan laporan, yang bertujuan untuk mendokumentasikan hasil penelitian sehingga dapat digunakan sebagai referensi oleh pembaca maupun menjadi dasar bagi penelitian lanjutan di masa mendatang.

V.KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian, dapat ditarik beberapa kesimpulan sebagai berikut.

1. Sistem *backend* Duri Pintar berhasil dikembangkan menggunakan *framework* Flask dengan implementasi fitur utama berupa autentikasi pengguna, prediksi penyakit daun durian, klasifikasi varietas durian, integrasi data cuaca, serta pencatatan riwayat pemindaian. Seluruh fitur tersebut telah dirancang sesuai kebutuhan pengguna dan skenario penggunaan pada aplikasi.
2. Integrasi model *machine learning* berbasis arsitektur *Xception* ke dalam *backend* telah berjalan dengan baik. Model mampu melakukan proses inferensi gambar melalui *endpoint* yang disediakan, dan menghasilkan prediksi penyakit maupun jenis durian lengkap dengan nilai *confidence* sehingga mendukung proses deteksi secara otomatis.
3. Pengujian fungsional dan pengujian integrasi menunjukkan bahwa seluruh *endpoint* berfungsi sesuai spesifikasi dan mampu menghasilkan respons JSON yang konsisten. Pengujian menggunakan Postman memastikan bahwa alur permintaan dan respons berjalan dengan baik tanpa *error* kritis, sehingga *backend* dinyatakan stabil untuk diintegrasikan dengan aplikasi *client*.
4. Pengujian performa melalui skenario *spike load* menunjukkan bahwa *backend* mampu menangani peningkatan jumlah permintaan secara tiba-tiba tanpa mengalami kegagalan permintaan. Waktu respons sempat meningkat pada beban puncak, tetapi sistem dapat kembali stabil setelah beban menurun, sehingga menunjukkan bahwa *backend* memiliki tingkat reliabilitas yang baik.
5. Proses pengembangan backend Duri Pintar serta dengan penyusunan dokumentasi sistem dan dokumentasi API sebagai bentuk pertanggungjawaban teknis penelitian. Dokumentasi diharapkan dapat menjadi acuan apabila sistem dikembangkan lebih lanjut. Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, sistem backend dinyatakan telah memenuhi tujuan penelitian.

5.2 Saran

Agar penelitian ini dapat berkembang lebih baik, beberapa saran yang dapat diberikan adalah sebagai berikut:

1. Disarankan untuk menambahkan mekanisme keamanan tambahan seperti enkripsi HTTPS, *rate limiting*, dan validasi input yang lebih ketat. Selain itu, penerapan *Continuous Integration*/*Continuous Deployment* (CI/CD) dapat membantu otomatisasi pembaruan sistem dan menjaga kestabilan layanan.

DAFTAR PUSTAKA

[1] N. Najira, E. Selviyanti, Y. B. Tobing, K. Kasmawati, R. Sianturi, and A. B. Suwardi, "Diversitas Kultivar tanaman Durian (Durio zubethinus Murr.) Ditinjau dari Karakter Morfologi," *J. Biol. Trop.*, vol. 20, no. 2, pp. 185–193, 2020, doi: 10.29303/jbt.v20i2.1871.

[2] Badan Pusat Statistik, "Produksi Tanaman Buah-buahan dan Sayuran Tahunan Menurut Provinsi dan Jenis Tanaman, 2024," bps.go.id. Accessed: Jan. 28, 2026. [Online]. Available: www.bps.go.id

[3] Badan Pusat Statistik, "Rata-Rata Harga Buah-Buahan di Kabupaten Ngawi," bps.go.id. Accessed: Jan. 28, 2026. [Online]. Available: www.bps.go.id

[4] W. Nengsih and S. Yulina, "Optimasi Model CNN untuk Identifikasi Jenis Bunga Berdasarkan spektrum Warna," *J. Komput. Terap.*, vol. 10, no. 1, pp. 57–66, 2024, doi: 10.35143/jkt.v10i1.6274.

[5] J. Al Gallenero and J. Villaverde, "Identification of Durian Leaf Disease Using Convolutional Neural Network," *2023 15th Int. Conf. Comput. Autom. Eng. ICCAE 2023*, pp. 172–177, 2023, doi: 10.1109/ICCAE56788.2023.10111159.

[6] Farhat Febrianto, "Implementasi Convolutional Neural Network Menggunakan Arsitektur Xception Untuk Identifikasi Penyakit Daun Pada Tanaman Durian," 2025.

[7] V. Tongsri, P. Songkumarn, and S. Sangchote, "Leaf spot characteristics of Phomopsis durionis on durian (durio Zibethinus Murray) and latent infection of the pathogen," *Acta Univ. Agric. Silvic. Mendelianae Brun.*, vol. 64, no. 1, pp. 185–193, 2016, doi: 10.11118/actaun201664010185.

[8] Angelina M. T. I. Sambi Ua *et al.*, "Penggunaan Bahasa Pemrograman Python Dalam Analisis Faktor Penyebab Kanker Paru-Paru," *J. Publ. Tek. Inform.*, vol. 2, no. 2, pp. 88–99, 2023, doi: 10.55606/jupti.v2i2.1742.

[9] A. J. Dhruv, R. Patel, and N. Doshi, "Python: The Most Advanced Programming Language for Computer Science Applications," no. Cesit 2020, pp. 292–299, 2021, doi: 10.5220/0010307902920299.

[10] D. F. Ningtyas and N. Setiyawati, “Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request,” *J. Janitra Inform. dan Sist. Inf.*, vol. 1, no. 1, pp. 19–34, 2021, doi: 10.25008/janitra.v1i1.120.

[11] N. Idris, C. F. Mohd Foozy, and P. Shamala, “A Generic Review of Web Technology: DJango and Flask,” *Int. J. Adv. Sci. Comput. Eng.*, vol. 2, no. 1, pp. 34–40, 2021, doi: 10.62527/ijasce.2.1.29.

[12] T. Purwanto, “Analisa Perbandingan Kinerja Rest Api Dengan Framework Flask, Laravel, Dan Express Js,” *Sci. Sacra J. Sains*, vol. 3, no. 4, pp. 49–55, 2023.

[13] I. R. D. Muhammad and I. V. Paputungan, “Development of Backend Server Based on REST API Architecture in E-Wallet Transfer System,” *J. Sains, Nalar, dan Apl. Teknol. Inf.*, vol. 3, no. 2, pp. 79–87, 2024, doi: 10.20885/snati.v3.i2.35.

[14] M. A. Novianto and S. Munir, “Analisis dan Implementasi Restful API guna Pengembangan Sistem Informasi Akademik pada Perguruan Tinggi,” *J. Inform. Terpadu*, vol. 8, no. 1, pp. 47–61, 2022, doi: 10.54914/jit.v8i1.409.

[15] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, “RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions,” *Appl. Sci.*, vol. 12, no. 9, 2022, doi: 10.3390/app12094369.

[16] M.B. Gigih Baskoro Ashari, “Implementasi Algoritma Convolutional Neural Network untuk Meningkatkan Identifikasi Penyakit Tanaman Durian,” *Jupiter Publ. Ilmu Keteknikan Ind. Tek. Elektro dan Inform.*, vol. 2, no. 4, pp. 162–172, 2024, doi: 10.61132/jupiter.v2i4.418.

[17] S. P. Sinde, B. Thakkalapally, M. Ramidi, and S. Veeramalla, “Continuous Integration and Deployment Automation in AWS Cloud Infrastructure,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 6, pp. 1305–1309, 2022, doi: 10.22214/ijraset.2022.44106.

[18] R. A. Pradipta, P. B. Wintoro, and D. Budiyanto, “Perancangan Pemodelan Basis Data Sistem Informasi Secara Konseptual Dan Logikal,” *J. Inform. dan Tek. Elektro Terap.*, vol. 10, no. 2, 2022, doi: 10.23960/jitet.v10i2.2541.

[19] M. Choina and M. Skublewska-Paszkowska, “Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine

libraries Analiza wydajności relacyjnych baz danych MySQL, PostgreSQL oraz Oracle z zastosowaniem bibliotek Doctrine," *J. Comput. Sci. Inst.*, vol. 24, no. July, pp. 250–257, 2022, [Online]. Available: https://ph.pollub.pl/index.php/jcsi/article/view/3000/2711%0Ahttps://ph.pollub.pl/index.php/jcsi/article/view/3000

[20] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, "Correction to: MongoDB Vs PostgreSQL: a comparative study on performance aspects (GeoInformatica, (2020), 10.1007/s10707-020-00407-w)," *Geoinformatica*, vol. 25, no. 1, pp. 241–242, 2021, doi: 10.1007/s10707-020-00424-9.

[21] A. D. Praba and M. Safitri, "Studi Perbandingan Performansi Antara Mysql Dan Postgresql," *J. Khatulistiwa Inform.*, vol. 8, no. 2, pp. 88–93, 2020, doi: 10.31294/jki.v8i2.8851.

[22] S. Pitriyani and R. Firdaus, "Pengembangan Data Base Terdistribusi untuk Aplikasi Cloud Computing," *Innov. J. Soc. Sci. Res.*, vol. 4, no. 3, pp. 15905–15917, 2024.

[23] A. Decan, T. Mens, P. R. Mazrae, and M. Golzadeh, "On the Use of GitHub Actions in Software Development Repositories," *Proc. - 2022 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2022*, pp. 235–245, 2022, doi: 10.1109/ICSME55016.2022.00029.

[24] D. Spinellis, Z. Kotti, and A. Mockus, "A Dataset for GitHub Repository Deduplication," *Proc. - 2020 IEEE/ACM 17th Int. Conf. Min. Softw. Repos. MSR 2020*, pp. 523–527, 2020, doi: 10.1145/3379597.3387496.

[25] D. Dewantoro, C. Kartiko, and F. Romadlon, "Implementasi Metodologi Kanban Dalam Pembuatan Aplikasi E-Commerce Pertanian Dengan Pendekatan Zachman Framework," *JOINTECS (Journal Inf. Technol. Comput. Sci.*, vol. 5, no. 2, p. 91, 2020, doi: 10.31328/jointecs.v5i2.1344.

[26] P. P. Kore, M. J. Lohar, M. T. Surve, and S. Jadhav, "API Testing Using Postman Tool," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 841–843, 2022, doi: 10.22214/ijraset.2022.48030.

[27] M. R. Rizanda, "Implementasi Convolutional Neural Network Berbasis Arsitektur Efficientnet-B0 Untuk Klasifikasi Varietas Buah Durian," 2025.

[28] S. J. Elroy, F. Nurdiyansyah, G. Priyandoko, and K. Kunci, “Klasifikasi Daun Durian Pada Citra Dalam Menentukan Jenis Menggunakan Convolutional Neutral Network,” pp. 56–62, 2024.

[29] Y. H. Natbais and A. B. S. Umbu, “Aplikasi Deteksi Penyakit pada Daun Tomat Berbasis Android Menggunakan Model Terlatih Tensorflow Lite,” *Teknotan*, vol. 17, no. 2, p. 83, 2023, doi: 10.24198/jt.vol17n2.1.

[30] G. Nathaniel, “Penerapan Framework Flask sebagai API Dalam Pengembangan Website Prediksi Kebakaran Hutan dan Lahan di Indonesia,” vol. 11, no. 6, pp. 6843–6847, 2024.

[31] I. G. M. Ariantara, I. Arwani, and W. H. N. Putra, “Penerapan REST API dalam Pengembangan Aplikasi Pemesanan RentalMobil berbasis Web dan Mobile (Studi Kasus: CV. Dwi Cipta Rent Car),” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 8, pp. 2569–2576, 2020.

[32] S. Andriansyah and Nurhasanah, “Perbandingan Performa Framework Laravel, Flask API Python, dan PHP Native untuk Aplikasi API pada Data AIS Polbeng,” *Konsep Desain Menentukan Hull Type, Mater. Dan Propulsi Unmanned Surf. Veh. Untuk Patroli Di Wil. Rokan Hiir Dengan Metod. Desicion Tree*, no. Lcm, pp. 478–486, 2020.

[33] J. A. Tradanarova, W. Hayuhardika, and N. Putra, “Pembuatan Sistem Informasi Pariwisata Kabupaten Kotawaringin Barat Kalimantan Tengah berbasis Aplikasi Mobile dengan Menggunakan Integrasi REST API,” *... Teknol. Inf. dan Ilmu ...*, vol. 7, no. 7, pp. 3296–3304, 2023, [Online]. Available:https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13019%0Ahttps://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/13019/5893

[34] D. Patel, “The Role of Amazon Web Services in Modern Cloud Architecture: Key Strategies for Scalable Deployment and Integration,” *Asian J. Comput. Sci. Eng.*, vol. 9, no. 4, pp. 1–9, 2024.

[35] C. Chandra1 *et al.*, “Perancangan dan Implementasi RESTful API untuk Aplikasi Mobile Pembelajaran Flora dan Fauna pada Google Cloud Platform,” vol. 4, no. 1, pp. 58–69, 2024, doi: 10.54259/satesi.v4i1.2850.

[36] C. Science, I. Hermadi, C. Science, Y. Nurhadryani, and C. Science,

"Analisis Uji Performa Aplikasi Dari Hasil Implementasi Refactoring Arsitektur Monolitik Ke Mikroservis dengan Decomposition dan Strangler Pattern," vol. 6, no. 3, pp. 189–203, 2023.

[37] Rudi Hartono, "Penerapan Kanban Model Sebagai Metode Perancangan Sistem Informasi (Studi Kasus: Pemetaan Sekolah SMA/K/MA Kota Tasikmalaya)," *J. Petik*, vol. 8, no. 1, pp. 27–34, 2022, [Online]. Available: https://journal.institutpendidikan.ac.id/index.php/petik/article/view/1247