IMPLEMENTASI *BACKEND* RESTFUL API MENGGUNAKAN EXPRESS.JS UNTUK INTEGRASI MODEL CNN PADA SISTEM MULTI-PLATFORM KLASIFIKASI SPESIES LEBAH MADU (INDO MALAYAN)

Oleh
Ghebi Armando

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

IMPLEMENTASI *BACKEND* RESTFUL API MENGGUNAKAN EXPRESS.JS UNTUK INTEGRASI MODEL CNN PADA SISTEM MULTI-PLATFORM KLASIFIKASI SPESIES LEBAH MADU (INDO MALAYAN)

Oleh

GHEBI ARMANDO

Pengembangan sistem klasifikasi citra berbasis kecerdasan buatan memerlukan sistem *backend* yang handal dan mampu mengelola komunikasi antar layanan, pengolahan data, serta penyediaan API secara stabil. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem *backend* berbasis Express.js yang terintegrasi dengan model CNN dalam bentuk RESTful API guna mendukung layanan klasifikasi citra lebah madu. Metode yang digunakan adalah *Rapid Application Development* (RAD) yang terealisasi dengan dua iterasi selama rentang empat bulan hingga ke tahap *cutover*. Sistem dikembangkan dengan menggunakan Express.js dengan integrasi Cloud Firestore, Firebase Storage, dan Google Cloud Platform sebagai lingkungan *deployment*. *Backend* yang dibangun menghasilkan 26 *endpoint* untuk mendukung autentikasi, manajemen informasi dan profil, dan klasifikasi citra. Hasil pengujian menunjukkan bahwa seluruh *endpoint* memperoleh validitas 100% pada 93 skenario pengujian fungsional. Pada pengujian performa, sistem mampu menangani hingga 44.400 *request* dengan *throughput* 75,07 *request*/detik dan rata-rata *response time* sebesar 89-165 ms (P90 < 300 ms) tanpa *error*. Hasil tersebut menunjukkan bahwa sistem *backend* yang dikembangkan memiliki performa yang responsif, stabil, dan mampu menangani variasi beban (*fixed load, ramp-up load, peak load,* dan *spike load*) melalui mekanisme *auto-scaling*. Dengan demikian, sistem *backend* yang dikembangkan dinyatakan layak digunakan sebagai layanan klasifikasi citra berbasis kecerdasan buatan yang dapat diakses melalui sistem *web* dan *mobile*.

Kata kunci: *Backend*, Express.js, RESTful API, GCP, Klasifikasi Citra

ABSTRACT

THE IMPLEMENTATION OF THE RESTFUL API BACKEND USES EXPRESS.JS FOR THE INTEGRATION OF THE CNN MODEL ON THE MULTI-PLATFORM SYSTEM OF CLASSIFICATION OF HONEY BEE SPECIES (INDO MALAYAN)

By

GHEBI ARMANDO

The development of an artificial intelligence based image classification system requires a reliable backend system that is able to manage communication between services, data processing, and API provision stably. This study aims to design and implement a Express.js-based backend system that is integrated with the CNN model in the form of a RESTful API to support honey bee image classification services. The method used is Rapid Application Development (RAD) which is realized in two iterations over a period of four months until the cutover stage. The system was developed using Express.js with the integration of Cloud Firestore, Firebase Storage, and Google Cloud Platform as a deployment environment. The built backend generates 26 endpoints to support authentication, information and profile management, and image classification. The test results show that the entire endpoint obtained 100% validity on 93 functional test scenarios. In performance testing, the system was able to handle up to 44,400 requests with a throughput of 75.07 requests/second and an average response time of 89-165 ms (P90 < 300 ms) without errors. These results show that the developed backend system has responsive, stable performance, and is able to handle load variations (*fixed load, ramp-up load, peak load, and spike load*) *through an auto-scaling mechanism. Thus, the developed backend system is declared feasible to be used as an artificial intelligence-based image classification service that can be accessed through web and mobile systems.*

Keyword: *Backend*, Express.js, RESTful API, GCP, *Image Classification*

Judul Skripsi	IMPLEMENTASI *BACKEND* RESTFUL API MENGGUNAKAN EXPRESS.JS UNTUK INTEGRASI MODEL CNN PADA SISTEM MULTI PLATFORM KLASIFIKASI SPESIES LEBAH MADU (INDO MALAYAN)
Nama Mahasiswa	: Ghebi Armando
Nomor Pokok Mahasiswa	: 2215061094
Program Studi	: Teknik Informatika
Jurusan	: Teknik Elektro
Fakultas	: Teknik

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Utama	Pembimbing Pendamping
Dr. Eng. Ir. Mardiana, S.T., M.T., I.P.M	Ir. Trisya Septiana, S.T., M.T., IPM
NIP. 197203161999032002	NIP. 199009212019032025

2. Mengetahui

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

MENGESAHKAN

1. Tim Penguji

Ketua : **Dr. Eng. Ir. Mardiana, S.T., M.T., I.P.M** .....

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

Penguji : **Yessi Mulyani, S. T., M. T.** .....

2. Dekan Fakultas Teknik

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

NIP. 196910302000001001

Tanggal Lulus Ujian Skripsi: 5 Juni 2026

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi berjudul "Implementasi *Backend* Restful Api Menggunakan Express.Js untuk Integrasi Model CNN pada Sistem Multi-Platform Klasifikasi Spesies Lebah Madu (Indo Malayan)" sepenuhnya merupakan hasil karya saya sendiri. Apabila di kemudian hari terbukti penyataan ini tidak benar, saya siap menerima sanksi sesuai dengan ketentuan hukum yang berlaku.

Bandar Lampung, 5 Juni 2026

Penulis

Ghebi Armando

NPM. 2215061094

RIWAYAT HIDUP

Penulis dilahirkan di Padang Bukit pada tanggal 8 September 2003, sebagai anak pertama dari pasangan Bapak Arman dan Ibu Reni Yuniarti. Penulis menyelesaikan pendidikan formal di SD Negeri 04 Enam Lingkung pada tahun 2016, kemudian melanjutkan ke SMP Negeri 1 Enam lingkung dan lulus pada tahun 2019, serta menamatkan pendidikan menengah atas di SMA Negeri 1 Enam Lingkung. Pada tahun 2022, penulis diterima sebagai mahasiswa Program Studi Teknik Informatika, Fakultas Teknik, Universitas Lampung, melalui jalur SBMPTN. Selama masa perkuliahan, penulis aktif berpartipasi dalam berbagai kegiatan, antara lain:

1. Mengikuti kegiatan Studi Independen Bersertifikat dari Kementerian Pendidikan dan Budaya pada mitra Bangkit Akademi pada tahun 2024.
2. Sebagai Asisten Laboratorium Teknik Digital yang bertanggung jawab dalam membantu proses perkuliahan para mahasiswa pada sesi praktikum pada 3 mata kuliah praktikum yang berbeda.
3. Mengikuti keanggotaan departemen Kominfo pada organisasi Badan Eksekutif Mahasiswa Universitas Lampung tahun 2025.

MOTTO

"Alam Takambang jadi Guru"

(Falsafah Hidup Minangkabau)

"Bukan karena hal-hal sulit kita tidak berani, tetapi karena kita tidak berani, maka hal itu menjadi sulit"

(Seneca)

"Hidup mengikuti dan menyesuaikan arus"

(Penulis)

"*Carpe diem, quam minimum credulo postero*"

(Horatius, Odes)

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 "Implementasi *Backend* Restful Api Menggunakan Express.Js untuk Integrasi Model CNN pada Sistem Multi-Platform Klasifikasi Spesies Lebah Madu (Indo Malayan)" 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. Orang tua dan adik semata wayang yang tidak pernah henti-hentinya memanjatkan do'a dan afirmasi yang menjadi tujuan untuk terus melangkah dan meraih hal-hal baik bagi kehidupan saya di masa depan.
2. Bapak Dr. 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 Universitas Lampung dan juga selaku Dosen Penguji yang memberikan kritik, saran, dan masukan yang membangun demi penyempurnaan hasil penelitian ini.
5. Bapak Puput Budi Wintoro, S. Kom. M. T. I., sebagai Dosen Pembimbing Akademik yang selalu memberikan motivasi dan arahan serta kritikan-kritikan penyemangat dari awal perjalanan perkuliahan hingga menyelesaikan skripsi ini.

6. Ibu Dr. Eng. Ir. Mardiana, S.T., M.T., IPM., selaku Dosen Pembimbin Utama, yang selalu memberikan arahan dan masukan yang menjadi motivasi penulis dalam menjalani perjalanan penelitian ini hingga menyelesaikan penulisan skripsi ini.
7. Ibu Ir. Trisya Septiana, S.T., M.T., IPM., selaku Dosen Pendamping yang selalu memberikan support dan meluangkan waktu serta tenaga dalam setiap sesi bimbingan, hingga dapat menyempurnakan dan memaksimalkan hasil penelitian ini.
8. Seluruh Dosen pada Program Studi Teknik Informatika, Fakultas Teknik, Universitas Lampung, yang tidak pernah lelah dalam memberikan dan membagikan ilmu, wawasan, dan pengalaman berharga selama masa perkuliahan.
9. Rekan satu tim penelitian, yaitu Arswendo Erza Sadewa dan Leo Fetri Hendli, atas kerja sama dan semangatnya dalam menjalani penelitian dan pengembangan sistem BeeVra.
10. Teruntuk sahabat COBRUT yaitu Edo, Ando, dan Melda yang tidak pernah henti-hentinya membersamai dan men-*support* penulis dari awal perkuliahan hingga bisa sampai dititik akhir penelitian ini.
11. Kak Aris dan rekan-rekan Asisten Laboratorium Teknik Digital, yang menjadi keluarga kedua di perantauan ini, selalu membagikan kehangatan, keceriaan pelajaran-pelajaran yang berharga selama beraktivitas di Laboratorium Teknik Digital.
12. Teman-teman PSTI B, serta rekan-rekan di Jurusan Teknik Elektro 2022 yang tidak dapat penulis sebutkan satu per satu, atas kebersamaan, dukungan, dan semangat yang diberikan selama masa perkuliahan hingga penyusunan skripsi ini terselesaikan.

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 *backend developer* dan komputasi. 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, 05 Juni 2026

Penulis

Ghebi Armando

NPM. 2215061094

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

I. PENDAHULUAN

1.1 Latar Belakang

Lebah madu (Apis spp.) memiliki peran penting dalam ekosistem melalui proses penyerbukan dan kontribusinya pada sektor ekonomi melalui produksi madu. Permintaan terhadap produk madu lokal di Indonseia semakin meningkat seiring berjalannya waktu, namun upaya dalam budidaya lebah madu ini sering menghadapi kendala akibat kesalahan identifikasi jenis lebah. Identifikasi manual berbasis morfologi membutuhkan keahlian entomologi yang tidak selalu tersedia, sehingga berpotensi menimbulkan bias dan ketidakefisienan [1].

Perkembangan teknologi *deep learning* memberikan alternatif yang lebih akurat dan efisien dalam menyelesaikan permasalahan ini. Pada penelitian [2], peneliti menggunakan algoritma *Convolutional Neural Network* (CNN) dan beberapa arsitektur modern lainnya, berhasil mengklasifikasikan subspesies lebah madu dari gambar sayap lebah madu dengan hasil akurasi mencapai 0,92. Penelitian lain dalam lingkup yang lebih kecil pada [1], menunjukkan bahwa CNN dapat membedakan tiga jenis lebah madu dengan akurasi tinggi, menunjukkan bahwa metode serupa relevan untuk lingkungan penelitian di Indonesia.

Penelitian mengenai klasifikasi lebah madu saat ini sebagian besar masih berfokus pada tahap pengembangan dan pengujian model tanpa melanjutkannya hingga ke tahap implementasi sistem yang dapat digunakan oleh masyarakat luas. Padahal,

pengembangan sistem yang terintegrasi melalui *backend* diperlukan untuk menjembatani model dengan aplikasi nyata. *Backend* berperan penting dalam menyediakan layanan *Application Programming Interface* (API) yang menjembatani model pembelajaran mesin dengan aplikasi pengguna, sehingga memungkinkan integrasi sistem berjalan secara aman, cepat, dan terukur [3].

Penerapan sistem *backend* yang terintegrasi dengan teknologi *machine learning* pada sistem klasifikasi citra menghadapi beberapa tantangan, antara lain latensi saat melakukan inferensi, kemampuan sistem dalam menangani permintaan yang besar, serta efisiensi penggunaan sumber daya. Studi pada [4], menunjukkan bahwa pemilihan *framework* model *serving* berpengaruh signifikan terhadap kinerja sistem, baik dari segi latensi maupun biaya operasional. Dengan demikian, sistem *backend* yang tepat menjadi faktor penentu keberhasilan implementasi sistem klasifikasi berbasis CNN.

Aspek keamanan juga menjadi perhatian utama pada *backend*, dengan berperan dalam melayani permintaan inferensi citra, sistem *backend* harus mampu mengantisipasi potensi serangan, mulai dari input yang tidak valid hingga serangan *denial of service*. Penelitian [5], menekankan perlunya penerapan mekanisme autentikasi, validasi input, serta pembatasan akses pada sistem berbasis *machine learning* untuk memastikan keandalan dan keamanan layanan.

Sistem *backend* dalam penerapannya juga harus mampu mendukung inferensi cepat dan responsif pada konteks aplikasi *real-time*. Hal ini diungkapkan pada penelitian [6], yang menunjukkan bahwa pengolahan inferensi pada sistem *streaming* memerlukan pendekatan khusus seperti *batching* dan pemisahan layanan *serving* dari API (*Application Programming Interface*) *gateway*. Hal ini menuntut desain arsitektur *backend* yang memperhatikan ketersediaan layanan dengan latensi minimal.

Di Indonesia, penggunaan Express.js dalam pengembangan *backend* terbukti efektif untuk membangun RESTful API yang menghubungkan aplikasi dengan model pembelajaran mesin. Penelitian [7], menunjukkan bahwa Express.js mampu memberikan fleksibilitas tinggi dalam integrasi sistem berbasis kecerdasan buatan. Dalam konteks pengembangan cepat dan adaptif, metode *Rapid Application Development* (RAD) menjadi pendekatan yang tepat. RAD menekankan iterasi cepat melalui siklus *planning*, *design workshop*, *construction*, dan *cutover* dengan keterlibatan aktif pengguna pada setiap tahap. Studi pada [8], membuktikan bahwa penerapan RAD mempercepat waktu dan meningkatkan efektivitas pengembangan sistem berbasis mobile tanpa mengorbankan kualitas.

Berdasarkan uraian tersebut, penelitian ini berfokus pada implementasi *backend* RESTful API menggunakan Express.js yang diintegrasikan dengan model CNN untuk klasifikasi jenis lebah madu. Pengembangan sistem dilakukan dengan metodologi RAD untuk memastikan proses pengembangan berlangsung cepat, iteratif, dan terarah dengan umpan balik pengguna secara berkelanjutan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana merancang arsitektur *backend* yang mampu mengintegrasikan model klasifikasi citra berbasis kecerdasan buatan ke dalam sebuah layanan digital yang terstruktur dan dapat diakses melalui API?
2. Bagaimana merancang dan mengimplementasikan *backend* berbasis Express.js yang terintegrasi dengan model CNN untuk klasifikasi jenis lebah madu?
3. Bagaimana *backend* dapat menyediakan layanan RESTful API yang aman,dan responsif untuk mendukung aplikasi *web* maupun *mobile*?

4. Bagaimana metodologi RAD dapat diterapkan dalam proses pengembangan *backend* untuk memastikan sistem terbangun secara terstruktur, adaptif, dan sesuai kebutuhan pengguna?

1.3 Batasan Masalah

Agar penelitian ini lebih terfokus dan terarah, maka ditetapkan batasan masalah sebagai berikut:

1. Penelitian ini hanya berfokus pada implementasi *backend* berupa RESTful API menggunakan *framework* Express.js.
2. Penerapan sistem *backend* pada penelitian ini hanya difokuskan pada integrasi model CNN dan membangun lingkungan komunikasi antara pengguna dan sistem,
3. Lingkup evaluasi *backend* difokuskan pada aspek fungsionalitas, performa (latensi dan *response time* API), serta keamanan dasar (autentikasi dan validasi input).

1.4 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah ditetapkan, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang arsitektur *backend* yang mampu mengintegrasikan model klasifikasi citra berbasis kecerdasan buatan ke dalam layanan digital yang terstruktur dan dapat diakses melalui API.
2. Mengimplementasikan *backend* berbasis Express.js yang terintegrasi dengan model CNN untuk klasifikasi jenis lebah madu.
3. Mengembangkan RESTful API yang dapat diakses oleh aplikasi *web* dan *mobile* untuk mendukung identifikasi lebah madu secara real-time.
4. Menerapkan metodologi RAD dalam proses pengembangan sistem agar menghasilkan *backend* yang terstruktur, efisien, dan sesuai kebutuhan.

1.5 Manfaat Penelitian

Berdasarkan tujuan penelitian diatas diharapkan dari penelitian ini memberikan manfaat yang meliputi:

1. **Manfaat Akademis**

   Penelitian ini diharapkan dapat memberikan kontribusi dalam bidang rekayasa perangkat lunak, khususnya mengenai implementasi *backend* yang mengintegrasikan model pembelajaran mesin ke dalam RESTful API.

2. **Manfaat Praktis**

   Sistem yang dikembangkan diharapkan menjadi sarana edukasi akademik dalam keanekaragaman lebah madu di zona Indo-Malayan.

3. **Manfaat Teknologis**

   Penelitian ini diharapkan dapat memperlihatkan bagaimana penerapan Express.js sebagai *framework backend* dapat mendukung integrasi model CNN dan layanan API yang siap digunakan oleh aplikasi *web* maupun *mobile*.

1.6 Sistematika Penulisan

Untuk memudahkan pemahaman, laporan penelitian ini disusun dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Menyajikan teori-teori yang mendukung penelitian, termasuk kajian mengenai CNN, *backend* development, RESTful API, Express.js, serta metodologi RAD, beserta hasil penelitian sebelumnya yang relevan.

BAB III METODOLOGI PENELITIAN

Membahas rancangan penelitian, arsitektur sistem, tahapan implementasi *backend*, serta penerapan RAD dalam proses pengembangan.

BAB IV HASIL DAN PEMBAHASAN

Menyajikan hasil implementasi *backend*, integrasi model CNN, pengujian fungsionalitas API, evaluasi performa, serta analisis kelebihan dan keterbatasan sistem.

BAB V KESIMPULAN DAN SARAN

Memuat kesimpulan dari penelitian yang telah dilakukan serta memberikan saran untuk pengembangan lebih lanjut.

II. TINJAUAN PUSTAKA

2.1 Sistem Multi-Platform

Sistem multi-platform merupakan sebuah pendekatan pada proses *software development* yang memungkinkan aplikasi berjalan pada berbagai *operation systems* dengan satu basis kode utama. Konsep pendekatan ini seiring berjalannya waktu menjadi penting dalam kebutuhan integrasi pada antara aplikasi *web* dan *mobile*, terkhusus pada pengembangan sistem yang terintegrasi dengan *machine learning* model, yang mengedepankan akses langsung dari pengguna ke antarmuka sistem secara langsung dan *real time*. Pada studi [9], pendekatan multi-platform seperti *Progressive Web Apps* (PWA) dan React Native mampu menekan akomodasi biaya pengembangan hingga 40%, hal ini dikarenakan penggunaan satu basis kode utama sebagai media integrasi dua platform utama yaitu, Android dan iOS.

Terlepas dari sisi efisiensi yang diberikan, sistem multi-platform ini juga mendukung interoperabilitas antara antarmuka pengguna serta *backend* melalui komunikasi API yang konsisten satu sama lainnya. Penelitian [10], peneliti memaparkan bahwa arsitektur multi-platform yang diintegrasikan dengan *backend* dapat meningkatkan *availibility* (ketersediaan layanan) serta menurunkan latensi komunikasi antar platform tersebut. Sistem ini dapat menjadi solusi yang relevan pada aplikasi klasifikasi citra dengan model *machine learning*, yang memerlukan *response time* yang rendah dan sinkronisasi data yang tinggi.

Pendekatan multi-platform dari sisi teknis juga memanfaatkan teknologi *Backend* as a *Service* (BaaS) guna menyederhanakan proses integrasi antar platform *frontend* dengan *server*. Penelitian lainnya pada [11], memaparkan bahwa hasil yang didapatkan dapa penggunaan arsitektur BaaS ini pada *framework* lintas platform seperti Flutter dan Firebase diketahui mampu mempercepat pengembangan aplikasi *e-learning* berbasis *Artificial Intelligence* (AI) hingga 30% dibanting metode konvensional yang biasa digunakan. Hal tersebut menegaskan bahwa konsep multi-platform ini tidak hanya unggul dalam menekankan kemudahan akses yang diberikan, namun juga pada efisiensi yang diberikan khususnya pada siklus *software development*.

Dalam konteks penelitian ini, konsep multi-platform menjadi pondasi penting untuk mengintegrasikan *backend* RESTful API dengan antarmuka *mobile* dan *web*, sehingga pengguna dapat berinteraksi secara langsung dengan sistem serta memperoleh hasil klasifikasi spesies lebah madu secara *real-time* dengan kinerja yang cepat dan efisien.

2.2 Arsitektur *Backend*

Backend menjadi komponen utama dalam arsitektur sistem multi-platform dengan tupoksi kerja pada logika bisnis, pengelolaan data, hingga komunikasi antar klien dan *server*. Dalam konteks sistem klasifikasi citra berbasis model CNN, *backend* bukan sekedar pembangun dan penyedia API, namun juga sebagai media penghubung antara model machine learning yang digunakan dengan antarmuka pengguna. Pada penelitian [12], menunjukkan bahwa arsitektur *backend* modern cenderung menggunakan pendekatan *microservices* dalam mendukung skalabilitas dan fleksibilitas sistem yang dibangun, terutama dalam melayani aplikasi multi-platform yang membutuhkan respons cepat dan konsisten.

Selain skalabilitas, aspek keamanan juga menjadi elemen penting dalam perancangan *backend*. Hasil studi [13], mengungkapkan bahwa arsitektur *cloud-native architecture* dengan kontrol akses yang terdistribusi mampu mengurangi risiko kebocoran data dan

meningkatkan efisiensi autentikasi pengguna hingga 25%. Temuan tersebut relevan dalam konteks sistem klasifikasi spesies lebah madu, mengingat *backend* bertanggung jawab atas pemrosesan serta penyimpanan data citra pengguna yang diunggah melalui aplikasi web maupun mobile.

Dengan demikian, perancangan arsitektur *backend* dalam sistem multi-platform tidak hanya terfokus pada konekstivitas dan komunikasi antar sisi, namun juga pada hal efisiensi kerja, keamanan sistem, serta skalabilitas yang mumpuni agar sistem siap diimplementasikan secara produksi.

2.3 Express.js

Express.js merupakan sebuah *framework* (kerangka kerja) untuk Node.js yang sangat menyederhanakan proses pembuatan *server* dan API pada pengembangan *backend* modern yang memanfaatkan ekosistem JavaScript di dalamnya. *Framework* Express.js ini juga memberikan kemudahan dalam pengorganisasian fungsionalitas aplikasi yang dikembangkan dengan memanfaatkan fungsi *middleware* dan *routing*, sehingga menambah *utility* yang bermanfaat ke objek HTTP dari Node.js dan memfasilitasi proses *rendering* yang dinamis serta mendefinisikan standar ekstensi yang mudah diterapkan [14].

Dalam konteks integrasi model *machine learning*, Express.JS ini sering digunakan sebagai penghubung antara model dan antarmuka pengguna. Penerapan Express.js dalam pengembangan sistem juga mendukung penggunaan REST API sebagai standar komunikasi multi-platform, sehingga memungkinkan model CNN sebagai *machine learning* model pada penelitian ini untuk dikonsumsi oleh aplikasi *web* maupun *mobile*. Selain keunggulan pada performanya, aspek keamanan juga menjadi *concern* utama dalam pengembangan API menggunakan Express.js.

Secara keseluruhan *framework* Express.js ini menjadi pilihan yang relevan dan efektif dalam mengembangkan *backend* pada sistem klasifikasi citra berbasis model CNN. Dengan kemudahan integrasi yang diberikan dengan layanan *cloud* seperti Google Cloud, dan fleksibilitas dalam penerapan API, serta skalabilitas yang baik dalam arsitektur multi-platform menjadikan *framework* ini selaras dengan kebutuhan penelitian ini. Pada studi [15], menunjukkan bahwa performa Express.js pada konteks integrasi model AI lebih unggul dibanding *framework* sejenis seperti Koa atau Hapi dalam *throughput* dan waktu responnya. Dengan demikian, penggunaan Express.js ini dinilai tepat sebagai pondasi arsitektur *backend* guna mendukung terciptanya sistem klasifikasi lebah madu yang efektif dan terukur dengan integrasi pada model CNN di dalamnya.

2.4 *Application Programming Interface (API)*

Application Programming Interface (API) merupakan antarmuka pemrograman yang berfungsi sebagai jembatan antara berbagai komponen perangkat lunak, sehingga sebuah aplikasi dapat menggunakan fungsionalitas dari komponen lain tanpa perlu mengetahui detail implementasinya. API ini juga memberikan abstraksi logis terhadap suatu masalah, yang dapat memberikan kemudahan dalam penggunaan ulang sebuah kode, serta memungkinkan bagi pengembang untuk membangun aplikasi yang modular dan terintegrasi [16]. Dalam pengembangan perangkat lunak, API menjadi elemen fundamental guna memastikan interoperabilitas antara berbagai komponen, seperti *backend*, *frontend*, serta layanan pihak ketiga yang digunakan. Dengan memanfaatkan API ini, pengembang dapat mengakses data atau fitur tertentu dari sistem lain menggunakan protokol dan format standar seperti HTTP, JSON, dan format protokol lainnya.

Pada penelitian [17], menunjukkan bahwa API bukan hanya digunakan sebagai alat komunikasi teknis, namun juga dapat digunakan sebagai infrastruktur penting dalam membangun suatu ekosistem digital yang kolaboratif dan juga aman. API dalam proses

pengembangan mendukung integrasi multi-platform dengan menyediakan *endpoint* yang terdefinisi secara eksplisit untuk operasi tertentu, seperti GET (pengambilan data), POST (pengiriman data baru), PUT/PATCH (pembaruan data), dan DELETE (penghapusan data). Hal ini menjadikan API berperan penting dalam menjaga konsistensi serta efisiensi komunikasi antar sistem dalam arsitektur berbasis layanan.

Peran API ini tidak hanya dalam lingkup yang kecil tersebut, namun peran API semakin signifikan dalam integrasi pada AI dan ML ke dalam berbagai platform. Dari hasil studi [18], peneliti menjabarkan bahwa API memungkinkan model AI di deploy sebagai layanan yang dapat diakses secara publik hingga global. Pendekatan tersebut mengubah cara model AI diimplementasikan pada sisi industri, dari yang semula bersifat lokal menjadi berbasis layanan (*AI as a Service*). Dalam arsitektur integrasi ini API berperan sebagai *middleware* yang menghubungkan model dengan berbagai aplikasi, dalam konteks penelitian ini yaitu sistem multi-platform.

2.5 RESTful API

Representational State Transfer (REST) API adalah salah satu arsitektur komunikasi berbasis HTTP (*HyperText Transfer Protocol*) yang kerap diterapkan dalam membangun sistem terdistribusi, yang mana sumber daya diidentifikasi melalui URI (*Uniform Resource Identifier*). Melalui model komunikasi berbasis HTTP ini, memungkinkan *server* untuk berkomunikasi dengan klien baik *web* maupun *mobile* melalui metode komunikasi seperti GET, POST, PUT, dan DELETE [19]. Pendekatan ini meningkatkan fleksibilitas dalam integrasi multi-platform, hal ini dikarenakan setiap klien hanya perlu mengirim permintaan dan menerima respons dari *server* tanpa mengetahui detail proses dari sisi *backend*. Dengan menyediakan struktur yang sederhana dan interoperabilitas tersebut menjadikan arsitektur ini banyak digunakan pada pengembangan sistem.

Arsitektur RESTful API ini menjadi pilihan dalam penelitian ini yang berperan sebagai penghubung atau media integrasi antara aplikasi klien dengan sistem *backend* yang di dalamnya juga mengelola model CNN yang bangun untuk klasifikasi citra lebah madu. Dengan memanfaatkan RESTful API, proses inferensi model dilakukan melalui *endpoint* tertentu, dengan tahapan klien mengunggah atau menangkap citra lebah madu secara langsung, sehingga *backend* dapat memprosesnya dan mengembalikan hasil klasifikasi lebah madu yang didapat.

Aspek keamanan menjadi poin penting dalam penerapan arsitektur RESTful API ini. Dari hasil penelitian [20], peneliti memaparkan bahwa penerapan *JSON Web Token* (JWT) dengan algoritma HMAC-SHA512 terbukti efektif dalam menjaga keamanan sesi pengguna serta memastikan proses autentitkasi yang aman antara *server* dan klien. Selain pada sisi autentikasi yang menjadi gerbang awal dari suatu sistem, praktik keamanan lain seperti pembatasan jumlah permintaan (*rate limiting*), enkripsi *transport layer* (HTTP/TLS), dan validasi data input juga menjadi hal penting yang perlu dipertimbangkan penerapannya dalam pengembangan *backend* menggunakan arsitektur RESTful API ini.

Dengan fungsi utama sebagai media komunikasi antara sisi *client* dan *server*, serta komponen keamanan yang disediakan oleh RESTful API dari berbagai temuan yang telah dilakukan. RESTful API menjadi pilihan penting dalam penelitian, guna mengelola komunikasi sistem berbasis Express.js dengan model CNN secara terstruktur, aman, dan dapat diakses pada sistem multi-platform. Kolaborasi dengan beberapa konfigurasi seperti JWT, validasi input, serta optimisasi performa API menjadi fokus utama agar sistem dapat beroperasi dengan baik dan sesuai skenario penggunaan nyata.

2.6 Sistem Klasifikasi Citra

Sistem klasifikasi citra merupakan salah satu cabang utama dalam bidang *computer vision* yang ditujukan untuk mengenali, mengelompokkan, serta mengklasifikasikan objek yang terdapat pada citra digital ke dalam kategori tertentu yang telah ditetapkan. Proses tersebut mencakup beberapa tahapan utama, meliputi akuisisi data, pre-processing citra, ekstraksi fitur, pelatihan model, dan evaluasi hasil klasifikasi yang didapatkan. Dengan perkembangan yang sangat pesat pada teknologi *deep learning* ini, memberikan dampak kemajuan yang pesat pada performa sistem klasifikasi citra, terutama pada penggunaan salah satu arsitektur yang banyak digunakan pada tujuan tersebut yaitu *Convolutional Neural Network* (CNN), dengan ketangkasan dalam mengekstraksi fitur secara otomatis dari data gambar yang ditangkap tanpa memerlukan rekayasa fitur manual.

Pada penelitian [21], memaparkan bahwa penggunaan arsitektur ResNet50 dan VGG16 dalam pengklasifikasian jenis kumbang di Indonesia menghasilkan tingkat akurasi mencapai 93% untuk model terbaik yang digunakan. Dengan mengandalkan dataset lokal yang dibantu latar belakang dan kondisi pencahayaan yang bervariasi memberikan tantangan praktis yang nantinya juga tidak bisa dipungkiri juga muncul dalam penelitian ini.

Penerapan CNN dalam sistem klasifikasi citra lebah madu tidak hanya berperan dalam konteks akademik, namun juga berdampak pada nilai praktis di lapangan. Sistem ini memberikan kemudahan pada proses identifikasi yang cepat di kondisi lapangan atau alam bebas. Dengan demikian, pengembangan sistem klasifikasi citra berbasis CNN dalam penelitian tidak hanya berfokus pada peningkatan akurasi prediksi, namun juga pada efisiensi sistem yang diberikan dan tingkat aplikatif yang menjadi nilai tambah dengan sistem multi-platform yang dikembangkan.

2.7 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) merupakan suatu metode pengembangan sistem sekuensial linear dengan menekankan pada sebuah siklus pengembangan sistem dengan waktu yang relatif singkat, sehingga dapat menghemat waktu dan proses pengembangan sistem menjadi lebih cepat. Pengaplikasian metode RAD ini dalam perancangan sebua perangkat lunak dapat membuat pengembangan dan pemeliharaan sebuah sistem menjadi lebih efisien [22][23].

Gambar 1. Tahapan RAD [23]

Dalam praktiknya RAD melewati empat fase utama, dengan penekanan iteratif pada tahapan guna mencapai hasil maksimal dalam penelitian yang dilakukan. Berikut tahapan-tahapan yang menjadi acuan dalam pengembangan berbasis RAD ini, yaitu:

1. *Requirements Planning*

   Tahap ini merupakan proses awal yang berfokus pada pengumpulan dan analisis kebutuhan pengguna melalui observasi dan metode lain seperti wawancara serta studi literatur. Informasi yang dikumpulkan menghasilkan rumusan kebutuhan fungsional dan non-fungsional sistem, khususnya terkait layanan RESTful API, integrasi model CNN, kebutuhan keamanan, serta spesifikasi performa *backend* yang diperlukan untuk mendukung aplikasi multi-platform. Keluaran dari tahap ini

mencakup daftar kebutuhan sistem, ruang lingkup penelitian, dan perencanaan komponen pendukung pengembangan.

2. *User Design*

   Pada tahap ini dilakukan perancangan arsitektur sistem secara menyeluruh, termasuk diagram alur, rancangan antarmuka *backend*, serta struktur direktori pengembangan berbasis Express.js. Fase ini menekankan iterasi cepat melalui *prototyping* desain, yang memungkinkan pengembang dan calon pengguna mengevaluasi rancangan sebelum implementasi dilakukan. Perencanaan mekanisme komunikasi antara *backend*, model CNN, Firebase Storage, dan Firestore juga dirumuskan pada tahapan ini sehingga alur sistem dapat tergambar secara jelas.

3. *Construction*

   Fase *construction* merupakan tahap implementasi inti yang berlandaskan desain yang telah disepakati sebelumnya. Aktivitas utama meliputi pembangunan layanan RESTful API, integrasi *endpoint* dengan model CNN, penerapan autentikasi, penyimpanan data, serta pengujian unit terhadap setiap fungsi sistem. Proses pengembangan dan pengujian dilakukan secara paralel dan iteratif sehingga perbaikan dapat dilakukan dengan cepat berdasarkan umpan balik pengguna maupun hasil pengujian. Tahap ini menekankan efisiensi, percepatan iterasi, dan penyempurnaan berkelanjutan.

4. *Cutover/Deployment*

   Cutover/deployment merupakan tahapan final yang mencakup proses *deployment* sistem ke lingkungan produksi menggunakan layanan Google Cloud Platform. Tahap ini memastikan bahwa API yang dibangun berfungsi secara stabil, aman, dan siap digunakan oleh pengguna pada aplikasi *web* maupun *mobile*. Aktivitas yang dilakukan mencakup konfigurasi lingkungan produksi, pengujian akhir, optimisasi performa, dokumentasi API, serta penyiapan sistem untuk penggunaan *real-time* oleh *end user*.

Pada penelitian [24], menuturkan bahwa menerapkan metode RAD dalam pengembangan sistem klasifikasi citra jahe-jahean dari keluarga *Zingiberaceae* menggunakan model EfficientNetB0 dengan melakukan iterasi prototyping cepat pada tahap desain dan implementasi, memungkinkan sistem diuji langsung oleh *end user* guna memperbaiki akurasi dan kinerja antarmuka. Hasil penelitian tersebut menunjukkan bahwa meskipun akurasi model masih perlu ditingkatkan (48%), pendekatan RAD efektif untuk mempercepat integrasi model klasifikasi citra dengan aplikasi berbasis *web*.

Dalam konteks penelitian ini, metode RAD diadopsi guna mendukung pembangunan RESTful API *backend* yang menghubungkan model klasifikasi citra lebah madu berbasis CNN dengan aplikasi multi-platform. Pendekatan ini dipilih karena mampu mempercepat proses pengujian dan penyempurnaan API melalui tahapan yang iterative, yang memungkinkan setiap komponen yang meliputi *endpoint*, autentikasi, dan integrasi model dapat diuji dan disesuaikan secara cepat berdasarkan hasil pengujian sistem dan kebutuhan pengguna secara langsung, sehingga evaluasi sistem dapat dilakukan dengan berkala dan cepat.

2.8 *Black Box Testing*

Black box testing merupakan salah satu metode pengujian perangkat lunak yang hanya terfokus pada pengujian fungsionalitas perangkat lunak tanpa mempertimbangkan struktur internal atau kode program penyusunnya. Di mana metode ini sangat berpengaruh besar pada kondisi yang digunakan untuk memastikan bahwa perangkat lunak berfungsi sesuai spesifikasi dan menghasilkan *output* yang benar berdasarkan input tertentu berdasarkan susunan yang telah ditentukan, yang mana dari keseluruhan metode tersebut tentunya terlepas dari bagaimana kode program itu diproses. Pengujian ini bertujuan untuk mengidentifikasi berbagai jenis kesalahan, seperti fungsi yang salah atau tidak lengkap, kesalahan antarmuka, kesalahan dalam struktur data atau akses ke

database eksternal (jika ada), kesalahan kinerja, serta kesalahan dalam proses inisialisasi dan terminasi sistem [25].

Pengujian pada sistem *backend* ini merupakan suatu tahapan penting dalam siklus pengembangan perangkat lunak. Hal ini ditujukan untuk memastikan bahwa sistem yang dikembangkan berfungsi sesuai dengan kebutuhan pengguna dan spesifikasi yang telah ditetapkan. Pengujian sistem tidak hanya dilakukan setelah menyelesaikan semua kebutuhan sistem, namun pada kondisi pengembangan sistem *backend* pengujian dilakukan pada setiap pengembangan *endpoint* yang telah diselesaikan secara modular di setiap kebutuhan.

2.9 Postman

Postman merupakan sebuah alat dalam proses pengembangan API yang dirancang guna menyederhanakan proses pembuatan, pengujian, serta dokumentasi API sebuah sistem. Postman menyediakan antarmuka grafis yang intuitif dan aplikatif, sehingga memungkinkan pengembang untuk merancang, menguji, dan mendokumentasikan API secara lebih efisien. *Tool* ini telah banyak digunakan oleh pengembang perangkat, *software tester*, dan *DevOps Engineer*, karena kehandalannya dalam membantu mengotomatisasikan pengujian, meningkatlan kolaborasi, serta mempercepat alur kerja pengembangan API [26].

Pada sisi kapabilitas, Postman mendukung proses desain API melalui pembuatan permintaan dan respons, pendefinisian *endpoint*, serta penyusunan *mock server* yang memungkinakan untuk mensimulasikan perilaku setiap API yang dibangun. Fitur ini memberi ruang bagi pengembang dalam bereksperimen dengan berbagai struktur data tanpa memerlukan sistem *backend* yang sudah berfungsi penuh (*trial and error process*). Selain itu, Postman juga menyediakan kemampuan pengujian dan *debugging* secara *real-time*, hal ini memungkinkan pengembang mengirim permintaan dan menerima respons secara langsung pada antarmuka Postman. Kemampuan ini juga

disertai dengan pengujian otomatis, pengujian integrasi sistem, dan pengujian beban sekaligus di dalamnya [26].

Postman juga mendukung proses pembuatan dokumentasi API yang dapat dipublikasikan dan dibagikan kepada pengguna atau pengembang lain. Dokumentasi API ini meliputi penjelasan mendalam mengenai setiap *endpoint*, format permintaan dan respons, serta informasi penting lainnya dari setiap *endpoint*. Hal tersebut dapat membantu pengguna atau pengembang lain dalam memahami penggunaan API dengan tepat. Dengan dukungan tambahan yaitu pada aspek kolaborasi, Postman memungkinkan pengembang memantau perubahan API dari waktu ke waktu [26].

2.10 Google Cloud Platform

Google Cloud Platform (GCP) merupakan salah satu platform *cloud computing* paling populer yang didirikan oleh Google dengan kompleksitas layanan dan produk yang disediakan, seperti penyimpanan data, mesin virtual, pembelajaran mesin. Platform ini memungkinkan penggunanya untuk fokus pada logika aplikasi inti dan *prototype* aplikasi dengan cepat, sehingga dapat membuat model bisnis yang dibangun lebih fleksibel dan cepat. GCP juga menyediakan layanan *severless computing,* meliputi Cloud Functions, Cloud App Engine, Cloud Run, dan Google Kubernetes Engine [27].

Pada pengembangan sistem klasifikasi citra, GCP berperan penting dalam menyediakan infrastruktur yang mendukung pemrosesan data dalam skala besar serta implementasi model *machine learning* seperti CNN melalui layanan seperti AI Platform, Compute Engine, Cloud Run, Cloud Storage, dan Cloud Firestore. Dengan layanan-layanan ini, pengembang dapat melakukan pelatihan hingga *deployment* model CNN tanpa harus mengandalkan sumber data komputasi lokal yang terbatas.

Dengan kelengkapan layanan yang disediakan, GCP bisa diterapkan dalam beberapa komponen penting dalam penelitian ini seperti, penyimpana dataset citra, *deploy* model

CNN, *endpoint* RESTful API sebagai *host* dalam memanggil model, serta manajemen keamanan (autentikasi dan *Identity and Access Management* untuk pembatasan akses pengguna). Sehingga pemilihan GCP diyakini dapat mendukung skalabilitas sistem klasifikasi lebah madu karena pengguna bisa mengakses API secara publik atau global, dan infrastruktur *cloud* ini memiliki elastisitas dalam penyesuaian beban inferensi yang dibutuhkan dalam penelitian ini.

2.11 Google App Engine

Google App Engine atau dikenal dengan GAE merupakan salah satu produk GCP yang termasuk ke dalam *Platform as a Service* (PaaS). Produk ini tergolong ke dalam tipe *serverless* yang memungkinkan untuk digunakan dalam media *deploy* aplikasi *web* dan *mobile* tanpa harus mengelola server secara langsung. App Engine ini mendukung berbagai bahasa pemrograman popular dan juga menyediakan berbagai alat untuk pengembang. Dengan tipe *serverless* yang dibawa oleh App Engine ini, pengembang dapat fokus pada pengembangan dan pengelolaan kode, sementara App Engine yang menangani semua urusan infrastruktur dari sistem [27].

2.12 Penelitian Terdahulu

Penelitian mengenai pengembangan sistem klasifikasi citra berbasis model CNN dengan penerapan integrasi pada arsitektur RESTful API telah banyak diterapkan baik pada penelitian tingkat nasional maupun internasional. Penelitian-penelitian tersebut meberikan gambaran mengenai bagaimana model CNN yang merupakan salah satu hasil dari *machine learning* diintegrasikan ke dalam suatu sistem berbasis multi-platform baik *web* maupun *mobile* melalui arsitektur *backend* yang modern. Namun masih terdapat ruang penelitian yang masih bisa ditelusuri dan dikaji, khususnya pada optimalisasi desain *backend* pada klasifikasi citra, terspesifikasi pada identifikasi spesies lebah madu yang belum banyk dikaji oleh para peneliti sebelumnya.

Pada penelitian [28], peneliti mengkaji implementasi model CNN sebagai media untuk diagnosis penyakit kulit melalui platform *web* dengan integrasi RESTful API yang dikembangkan dengan menggunakan *framework* Flask dan Docker (*containerization*). Dengan sistem tersebut didapatkan hasil dari efektifitas model CNN dalam mengenali pola citra medis dengan mengembalikan output inferensi melalui permintaan pada komunikasi HTTP. Meskipun sama pada sisi penerapan integrasi RESTful API, namun penelitian ini terfokus pada penerapan pada domain Kesehatan dan belum mengkaji langsung pada klasifikasi citra di alam terbuka, seperti pada klasifikasi spesies lebah madu dengan kompleksitas visual yang jelas berbeda.

Penelitian lainnya pada [29], membahas mengenai integrasi model CNN algoritma Resnet101 dengan RESTful API yang dibangun dengan *framework* Flask yang berbasis Python serta teknologi Firebase sebagai *database* platform *mobile app* sistem identifikasi EEG gangguan spektrum autism. Didapatkan bahwasanya penelitian ini dengan penerapan integrasi CNN dengan RESTful API berhasil menciptakan sistem guna mengidentifikasi kondisi ASD dengan tingkat akurasi mencapai 98,7% (data pelatihan) dan 94,3% (data validasi) dengan hasil pengujian pada sisi *backend* yang menunjukkan rata-rata *delay* 100-238 ms dan latensi 50-133 ms, temuan ini memberikan hasil yang potensial dalam deteksi dini gangguan *neurodevelopmental* dengan efisiensi sistem yang dihasilkan tersebut.

Pada studi lainnya [30], pengembangan aplikasi rekomendasi destinasi wisata berbasis *machine learning* (TensorFlow) menggunakan Node.js dan Express.js menunjukkan bahwa Express.js mampu menangani request dan response dengan cepat sebagai platform *server-side*. *Framework* ini juga memberikan kinerja stabil untuk pengolahan data dalam jumlah besar serta fleksibilitas tinggi untuk pengembangan dan pembaruan di masa mendatang.

Dalam konteks keamanan, studi pada [31], menunjukkan bahwa penerapan autentikasi berbasis token JWT (JSON Web Token) serta mekanisme hashing kata sandi

menggunakan bcrypt pada arsitektur RESTful API mampu meningkatkan integritas dan keamanan sistem secara signifikan. Penelitian ini juga menekankan bahwa keamanan API tidak hanya berfokus pada autentikasi, tetapi juga mencakup perlindungan data, pengendalian akses, dan pengamanan *endpoint*. Temuan tersebut menjadi dasar penting bahwa perlunya penerapan praktik keamanan API untuk meminimalkan risiko pelanggaran data dan juga meningkatkan keandalan sistem.

Pada penelitian [32], mengkaji siklus pengembangan khusus pada bahasa pemrograman TypeScript dan Node.js dengan *framework* Express.js dan Vercel sebagai alat bantu. Hasil penelitian ini menunjukkan bahwasanya pengembangan sistem yang dilakukan dengan teknologi yang tidak begitu populer seperti Express.js, dapat menjadi alternatif yang baik karena memiliki keunggulan dalam hal fungsionalitas khususnya sampai pada tahap *deployment* yang sangat berdampak dalam pengembangan aplikasi baik *web* maupun *mobile*.

Penelitian lainnya [33], menggunakan Express.JS dan Node.JS dalam pengembangan RESTful API pada sistem informasi manajemen rekomendasi karyawan. Hasil penelitian menunjukkan bahwa Express.JS mampu mendukung pengembangan layanan *backend* yang efisien, modular, dan mudah diintegrasikan dengan sistem lain. Dengan hasil tersebut Express.JS menjadi pilihan yang relevan pada teknologi yang digunakan dalam pengembangan *backend* pada sistem klasifikasi citra jenis lebah madu.

Penelitian lain yang berfokus pada pengembangan *backend* dengan Node.js, Express.js, dan Google Cloud Platfom dimana pengembangan dilakukan hingga tahap *deployment*. Pada studi [34], memaparkan hasil kajian tersebut bahwasanya penerapan teknologi *microservices* dan *serverless* dari layanan Google Cloud Platform terbukti menjadi standar arsitektur baru di industri. Dengan teknologi *backend* Node.js dan Express.js yang digunakan dalam kolaborasi pada pengembangan sistem tersebut berdampak pada pengurangan biaya proyek mencapai setengah biaya biasanya, baik dari sisi

pengembangan, pemeliharaan, maupun *deployment*. Hal ini memberikan pendekatan pengembangan aplikasi yang lebih hemat, efisien, dan ramah bagi pengembang.

Pada hasil penelitian [35], yang mana penelitian ini berfokus pada penggunaan *database* NoSQL berbasis Cloud Firestrore melalui pemisahan struktur data agregasi untuk meningkatkan efisiensi biaya, waktu *response time*, dan ukuran respons. Penelitian ini menghasilkan metode terbaik dalam perangcangan arsitektur *database* terutama bagi pengembang aplikasi berbasis *cloud*.

Selain itu, dalam konteks penerapan teknologi Google Cloud Platform yang diintegrasikan dengan model CNN pada sistem berbasis *mobile*, berdasarkan temuan [36], memaparkan hasil yang menjanjikan pada industri peternakan khususnya peternakan telur, dengan hasil kolaborasi pada beberapa teknologi modern seperti CNN, GCP dan *mobile dev* menciptakan solusi yang efektif pada upaya monitoring dan pengambilan keputusan dala industri ini. Sehingga penerapan salah satu teknologi yaitu GCP pada penelitian ini menjadi solusi teknis yang dapat berdampak positif pada industri peternakan lainnya yaitu peternakan lebah madu dalam upaya pelestarian dan peningkatan produksi madu dengan sistem klasifikasi citra spesies lebah madu.

Dalam lingkup pengujian sistem [37], peneliti menerapkan pengujian otomatis berbasis spesifikasi dalam menguji hasil *endpoint* RESTful API. Hasil yang didapatkan dari penelitian dengan menggunakan pendekatan tersebut, dapat meningkatkan cakupan pengujian, efisiensi deteksi kesalahan, dan keandalan API apabila dibandingkan dengan pengujian manual. Hasil ini menunjukkan peningkatan yang signifikan dalam efektivitas dan efisiensi dalam pengujian API.

III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat

Penelitian ini dilaksanakan pada November 2025 sampai dengan Februari 2026, dimana penelitian ini bertempat di Suhita Bee Farm, Lembah Suhita, Kec. Teluk Betung Utara, Kota Bandarlampung, Lampung. Berikut adalah tabel 1 yang menunjukkan jadwal kegiatan penelitian yang dilakukan pada waktu dan tempat yang tertera.

Tabel 1 *Timeline* Penelitian

No.	Aktivitas	November 2025				Desember 2025				Januari 2026				Februari 2026			
		Minggu ke-															
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.	*Requirement Planning*																
2.	*User Design*																
3.	*Construction*																
4.	*Cutover*																
5.	Penulisan Laporan																

3.2 Alat Penelitian

Penelitian memiliki beberapa alat penunjang jalannya penelitian dan pengembangan sistem klasifikasi lebah ini, berupa perangkat lunak dan perangkat keras dengan. Berikut rincian alat penelitian yang digunakan yaitu:

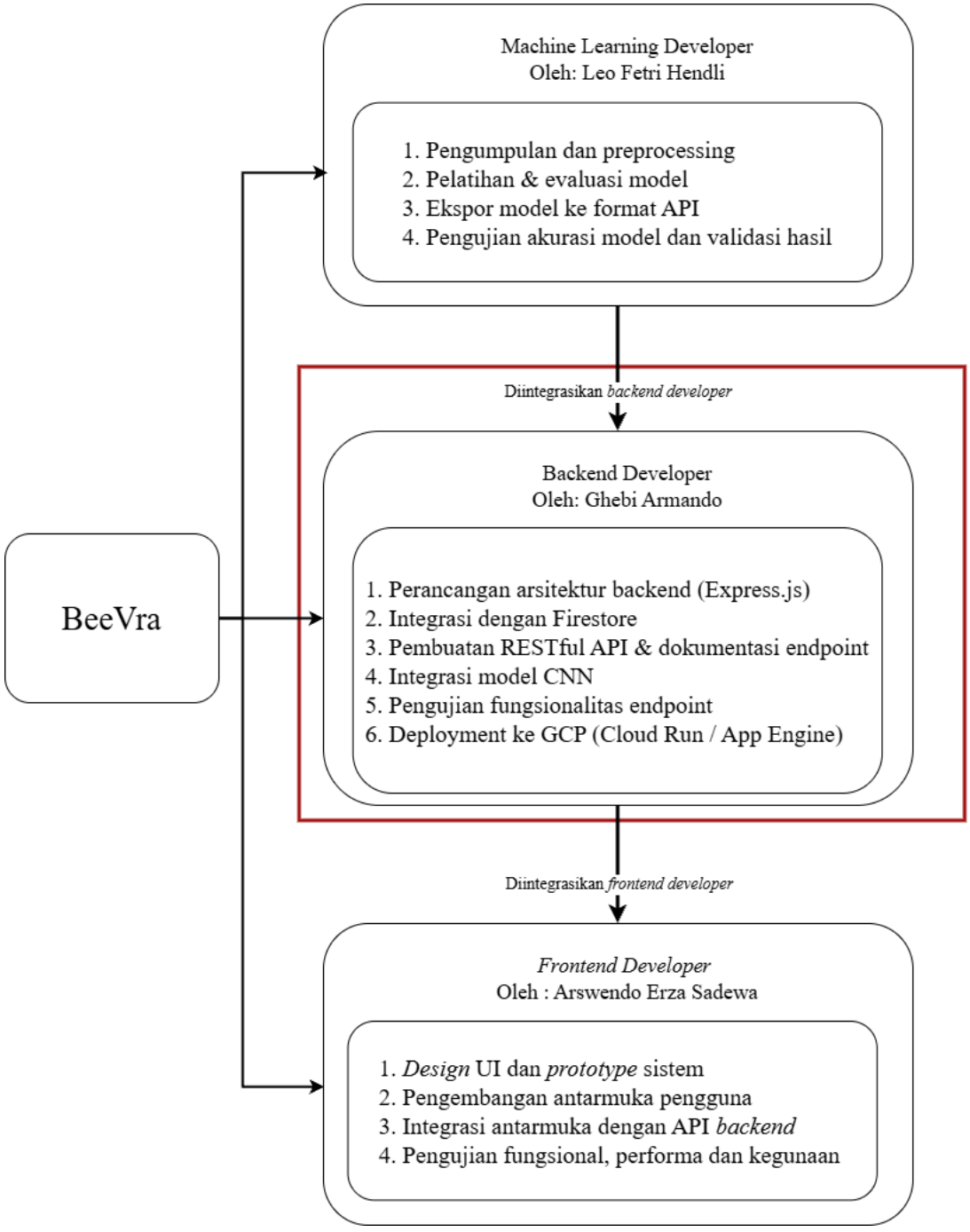
Tabel 2 Alat Penelitian

No.	Alat	Spesifikasi	Fungsi
1.	Laptop	Jenis: VivoBook_ASUSLaptop X415DAP M415DA	Perangkat keras yang digunakan dalam keseluruhan aktivitas pada penelitian dan pengembangan sistem ini.
		Processor: AMD Ryzen 3 3250U	
		Graphics: Radeon Graphics 4CPUs 2.6GHz	
		RAM & ROM: 8GB, Memory 512GB	
2.	*Operating System*	Windows 11	Sistem operasi utama yang digunakan sebagai otak proses pada perangkat keras.
3.	*Software IDE*	Visual Studi Code	Perangkat lunak IDE yang digunakan dalam membuat kode program pengembangan sistem.
4.	*Runtime Environment*	Node.JS	*Runtime Environment* Javascript yang digunakan untuk menjalankan *server* pengembangan pada sisi lokal (instalasi *library*, *dependencies*, hingga *production*)

Tabel 2 Alat Penelitian (lanjutan)

No.	Alat	Spesifikasi	Fungsi
5.	*Framework*	Express.JS (v 5.6.0)	*Framework* utama yang digunakan dalam membangun sistem *server*, pengelolaan *database*, dan merancangan alur komunikasi antar bagian.
6.	*Library Development*	• Multer (v 2.0.2) • Axios (v 1.13.2) • Resend (v 6.6.0) • Nodemon (v 3.1.11) • Dotenv (v 17.2.3) • jsonwebtoken (v 9.0.2) • Firebase-admin (v 13.6.0) • Bcryptjs (v 3.0.3) • Uuid (v 13.0.0)	*Library development* sebagai *tools* dan media yang digunakan untuk membangun sistem *backend*.
7.	*Version Control*	Git & Github	Perangkat lunak *version control* sebagai *repository* bersama tim.
8.	*Deployment Platform*	Google Cloud Platform	Layanan *cloud* yang dikelola oleh Google, digunakan sebagai layanan *deployment* sistem *backend* (Cloud App Engine)
9.	*Testing Tools*	Postman	Perangkat lunak yang digunakan sebagai pengujian fungsionalitas sistem *backend*
10.	*Database System*	Firebase	Layanan *cloud* sebagai pengelola *database* sistem seperti Cloud Firestore dan Firebase Storage.

3.3 Tim Penelitian



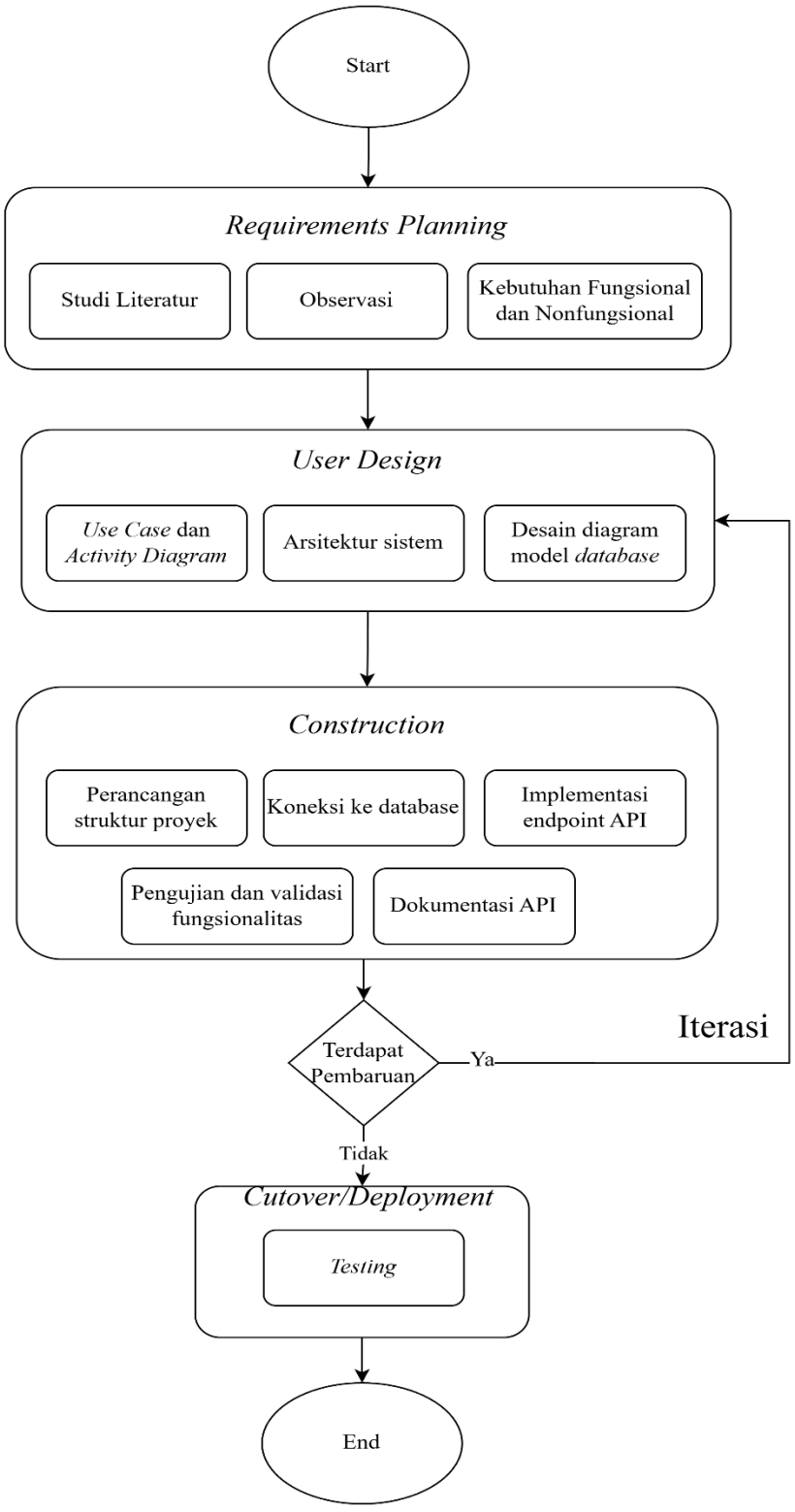
Gambar 2. Pembagian Tugas Tim Penelitian

Perancangan dan pengembangan sistem klasifikasi citra spesies lebah madu ini, dibagi ke dalam 3 tim penelitian dengan fokus dan pembagian tugas masing-masing. Tim tersebut terdiri dari, *Machine Learning Developer, Backend Developer,* dan *Frontend Developer* dengan pembagian tugas seperti pada gambar 3. Dengan pembagian tugas dan tanggung jawab tim yang jelas dan terstruktur, berguna dalam efektivitas pengembangan yang dilakukan tanpa tertanggu intervensi antar sesama tim apabila tidak ditetapkan tupoksi masing-masing tim.

Dengan pembagian tugas yang telah ditetapkan, penelitian ini ditekankan pada konteks sistem *backend* (dibatasi di dalam wilayah garis merah) dari aplikasi klasifikasi citra spesies lebah madu ini. Penelitian dan pengembangan pada lingkungan sistem *backend* ini meliputi beberapa tugas spesifik yang telah disusun dengan sistematis di dalamnya, yaitu perancangan arsitektur *backend* (menggunakan *framework* Express.js), integrasi sistem pada *database* Firestore, pembuatan RESTful API dan dokumentasi *endpoint* sebagai media komunikasi antara sistem dan pengguna, integrasi model CNN, pengujian fungsionalitas *endpoint* yang telah dibangun, dan terakhir peluncuran sistem *backend* (*deployment*) pada platform Google Cloud Platform dengan memanfaatkan layanan Cloud App Engine.

3.4 Tahapan Penelitian

Pada tahapan penelitian ini menerapkan pendekatan *Rapid Application Development* (RAD). Metode pendekatan ini digunakan sebagai fokus penelitian dan pengembangan pada aspek kecepatan pengembangan, interaksi dan komunikasi langsung dengan pengguna, serta aspek interatif yang diterapkan di dalamnya. Pendekatan ini terdiri atas beberapa tahapan, terlihat pada gambar 2 di bawah.



Gambar 3. Tahapan Penelitian

3.4.1 *Requirement Planning*

Tahapan *requirement planning* ini, dilakukan proses identifikasi secara menyuluruh dan komprehensif untuk setiap kebutuhan pengguna. Hal ini ditujukan untuk merumuskan spesifikasi kebutuhan sistem dalam sisi fungsional dan non-fungsional. Rumusan kebutuhan yang dilakukan menjadi landasan dalam pengembangan *backend* untuk sistem klasifikasi citra spesies lebah madu pada penelitian ini. Analisis kebutuhan ini dilakukan melalui studi literatur berlandaskan penelitian dan sistem serupa sebagai media pembanding dalam upaya pembaruan yang dilakukan. Kajian arsitektur juga dibutuhkan guna media integrasi model CNN sebagai layanan eksternal, serta eksplorasi pada sisi teknis (*framework* Express.js) sebagai arsitektur pengembangan *backend*.

Kebutuhan utama pengembangan *backend* pada penelitian ini juga ditentukan seperti pada layanan penerimaan data yang diproses model CNN setelahnya melalui RESTful API, mengelola autentikasi dan otorisasi pengguna, hingga memberikan respons hasil klasifikasi secara *real-time*. Selain kebutuhan fungsional di atas, kebutuhan non-fungsional juga tidak luput dari fokus pada tahapan ini, hal tersebut meliputi, kecepatan respons sistem, reliabilitas integrasi, skalabilitas layanan pada tahapan *deployment* di lingkungan Google Cloud Platform, serta aspek keamanan dari setiap data user yang tersimpan. Infrastruktur sistem yang digunakan juga ditentukan pada tahapan ini, seperti Cloud App Engine pada layanan Google Cloud Platform sebagai media hosting *backend* yang digunakan dan Cloud Firestore berbasis NoSQL yang digunakan sebagai penyimpanan basis data.

3.4.2 *User Design*

Tahapan *user design* pada penelitian ini difokuskan pada perancangan sistem *backend*, yang berperan sebagai pusat komunikasi antara aplikasi *web* dan *mobile*, basis data, serta layanan kecerdasan buatan berupa model CNN. Dengan adanya pembagian tugas pada untuk setiap anggota tim, peneliti bertanggung jawab penuh terhadap perancangan

dan pengembangan *backend* yang mengelola alur logika sistem, komunikasi antar layanan, manajemen data, serta aspek keamanan sistem. Perancangan yang dilakukan pada tahapan ini bertujuan untuk menghasilkan desain sistem *backend* yang modular, terstruktur, dan mudah diintegrasikan dengan model CNN serta platform *client-side* yaitu aplikasi *web* dan *mobile*. Perancangan tersebut meliputi perancangan arsitektur *backend* yang mendefinisikan hubungan dan alur komunikasi antara *client*, *backend*, *database*, dan layanan model CNN.

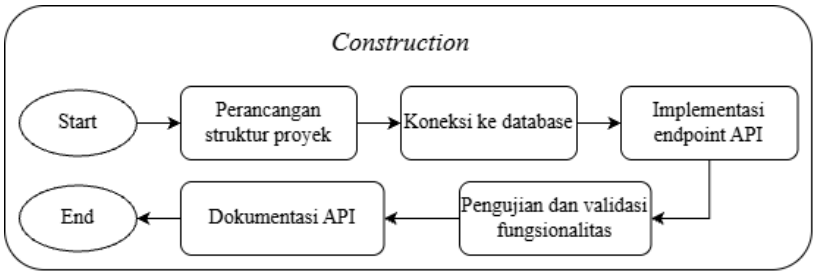
User design ini juga meliputi perancangan struktur direktori *backend* berbasis Express.JS yang mencakup komponen folder berupa *routes*, *controllers*, *middlewares*, dan *utils*. Pendekatan modular ini diterapkan untuk menjaga keteraturan kode, meningkatkan keterbacaan, serta memudahkan proses pengembangan dan pemeliharaan sistem di tahap selanjutnya. Selain itu, juga dirancangan mekanisme komunikasi *backend* dengan model CNN juga dirancangan pada tahapan ini yaitu menggunakan arsitektur RESTful API dengan komunikasi metode HTTP berupa, GET, POST, PUT, PATCH, dan DELETE. Sistem *backend* sebagai penghubung utama yang menerima input dari pengguna, memproses permintaan, serta meneruskan data ke layanan model CNN untuk proses klasifikasi citra.

Beberapa *endpoint* utama didefinisikan dalam tahap *user design*, antara lain *endpoint* klasifikasi citra pada *"/api/classify/"* yang digunakan untuk mengirimkan data citra ke model CNN, serta *endpoint* manajemen sistem seperti *"/api/auth/"*, *"/api/bee/"*, *"/api/admin/"*, dan *"/api/profile/"* yang digunakan untuk mengelola autentikasi, data pengguna, dan data sistem lainnya. Perancangan *endpoint* ini menjadi dasar implementasi *backend* pada tahap *construction*.

3.4.3 *Construction*

Tahapan *construction* pada penelitian ini merupakan tahap implementasi dari keseluruhan rancangan yang telah disusun pada tahapa *user design*. Pada tahapan ini,

rancangan arsitektur sistem, proses bisnis, serta desain komunikasi antar layanan direalisasikan ke dalam bentuk sistem *backend* yang dapat dijalankan dan diuji secara fungsional. Proses construction ini difokuskan pada pengembangan sisi *backend* dengan menggunakan *framework* Express.js yang terintegrasi dengan Cloud Firestore dan Firebase Storage sebagai layanan *database* sistem, serta penerapan RESTful API untuk komunikasi antar layanan sistem.



Gambar 4. Alur Kerja Tahapan *Construction*

Berdasarkan Gambar 4, proses pengembangan diawali dengan membangun struktur proyek *backend* sesuai dengan rancangan struktur folder pada tahap *user design*. Struktur proyek dibangun secara modular berbasis Express.js ke dalam beberapa folder utama berupa *routes*, *controllers*, *middlewares*, dan *utils*. Pendekatan modular ini diterapkan guna memastikan kemudahan dalam pengelolaan kode, meningkatkan *maintainabilitiy*, serta mendukung skalabilitas sistem dalam pengembangan selanjutnya.

Setelah struktur proyek dibangun, tahapan berikutnya adalah membangun koneksi ke *database* yaitu Cloud Firestore sebagai *database* NoSQL dan Firebase Storage sebagai media penyimpanan berkas citra atau gambar sistem. Koneksi ini memungkinkan sistem *backend* yang dibangun, dapat menyimpan dan mengelola data pengguna, metadata hasil klasifikasi, histori klasifikasi citra secara terpusat dan terstruktur.

Dengan konfigurasi utama yaitu struktur proyek telah dibangun dan sistem *backend* telah terkoneksi ke *database*, implementasi *endpoint* RESTful API dapat dilakukan

secara menyeluruh sesuai dengan perancangan pada tahap sebelumnya. Implementasi ini mencakup proses autentikasi dan otorisasi pengguna menggunakan teknologi *JSON Web Token* (JWT), pengelolaan data pengguna dan spesies lebah, serta pengelolaan data hasil klasifikasi citra. Selain itu, pada tahap *construction* ini, proses integrasi model CNN juga direalisasikan melalui *endpoint* khusus, sehingga sistem *backend* dapat menerima berkas citra dalam format PNG atau JPEG dari pengguna. Kemudian data citra yang diterima diteruskan ke layanan model CNN melalui komunikasi HTTP POST. Selanjutnya input model CNN diproses dan sistem *backend* menerima hasil klasifikasi citra dalam format JSON, untuk selanjutnya disimpan ke dalam *database* dan dikirim kembali ke pengguna sebagai respons sistem.

Tahapan *construction* ini tidak terbatas pada implementasi kode, tetapi juga mencakup proses verifikasi dan validasi internal terhadap sistem yang sedang dikembangkan. Pengujian pada tahapan ini difokuskan pada memastikan bahwa setiap komponen *backend* yang dibangun telah berfungsi sesuai dengan rancangan pada tahap *user design*. Pengujian dilakukan pada tingkat fungsional (*functional testing*) guna memastikan setiap *endpoint* REST API dapat menerima *request* dan menghasilkan respons yang sesuai dengan spesifikasinya. Selain itu, dilakukan pula i*ntegration testing* untuk memverifikasi komunikasi antar komponen internal sistem, seperti interaksi antara *controller*, *middleware*, serta koneksi ke Cloud Firestore dan Firebase Storage dapat berjalan. Selain itu, dilakukan *system testing* pada tahapan ini dalam lingkungan pengembangan untuk memastikan seluruh modul *backend* dapat berjalan secara penuh dan terpadu sebelum memasuki tahap *deployment*.

Aspek keamanan sistem juga diterapkan pada tahap ini, meliputi validasi input pengguna, pembatasan ukuran berkas citra yang diunggah pada proses klasifikasi, serta konfigurasi CORS (*Cross-Origin Resource Sharing*) untuk membatasi domain yang diizinkan mengakses layanan *backend*. Penanganan kesalahan (*error handling*) diimplementasikan melalui *middleware* khusus guna memberikan respons kesalahan

yang informatif dan konsisten, seperti pada kesalahan autentikasi, validasi input, maupun kegagalan layanan lainnya.

Sebagai tahap akhir dalam proses construction, dilakukan penyusunan dokumentasi API menggunakan *tools* pendukung seperti Postman Documentation. Dokumentasi ini bertujuan untuk memberikan gambaran yang jelas mengenai setiap *endpoint* REST API yang dibangun, sehingga dapat dipahami dan digunakan dengan baik oleh pengembang lain maupun pihak terkait dalam proses pengembangan dan integrasi sistem.

3.4.4 *Cutover*

Tahapan *cutover* menjadi tahapan transisi sistem dari lingkungan pengembangan menuju lingkungan produksi. Pada tahapan ini juga terdapat pengujian di tahap akhir, namun pengujian yang dilakukan dalam tahapan ini berbeda dengan tahapan *construction*. Yang mana pengujian pada tahapan *construction* berfokus pada verifikasi fungsionalitas dan integrasi internal selama proses pengembangan, sedangkan pada tahapan ini pengujian difokuskan pada validasi kesiapan operasional sistem secara menyeluruh setelah proses *deployment* dilakukan.

Tahapan ini dimulai dengan validasi menyeluruh pada setiap komponen *backend* yaitu pada setiap *endpoint* API yang dibangun, integrasi prediksi citra dengan model CNN, pengelolaan respons sistem, dan penanganan *error* serta komunikasi sistem dan *database* Firestore. Pengujian dilakukan menggunakan pendekatan *system testing* dan *integration testing* pada lingkungan produksi untuk memastikan stabilitas integrasi antar layanan serta konsistensi performa sistem pada berbagai kondisi dan variasi beban pengujian.

Apabila sistem telah dinyatakan stabil secara operasional dan fungsional, maka proses *deployment* atau tahap *production* pada lingkungan Google Cloud Platform siap dilakukan, dengan memanfaatkan layanan PaaS (*Platform as a Service*) yaitu Cloud

App Engine, pemilihan layanan ini dikarenakan layanan ini menerapkan arsitektur *serverless*, efisiensi biaya berbasis penggunaan, serta kemudahan proses *deployment* yang diberikan melalui file konfigurasi (app.yaml).

Konfigurasi dan yang dilakukan juga meliputi komunikasi dengan *database* yaitu Cloud Firestore. Jenis *database* ini juga dipilih karena menyedikaan mekanisme penyimpanan dara yang *real-time*, terstruktur dalam bentuk dokumen atau *koleksi*, serta kemudahan pada integrasi dengan *backend* Express.js. Pada saat *deployment* dilakukan, konfigurasi yang bersifat privasi, seperti *secret key* dan kredensial lainnya disimpan pada berkas (.env) guna menjaga keamanan sistem.

Setelah semua konfigurasi dan alur sistem telah masuk dan berjalan pada sisi *production*, maka sistem siap pakai tersebut dijadikan dasar pada proses evaluasi. Evaluasi dilakukan dengan menerapkan beberapa kriteria yang terfokus pada aktivitas *endpoint* API yang dibantu *tools* pengujian yaitu Postman. Tabel berikut menunjukkan kriteria evaluasi beserta target keberhasilan yang dibuthkan pada tahapan evaluasi ini.

Tabel 3 Kriteria Evaluasi Pengujian *Endpoint* API

No	Kriteria Evaluasi	Deskripsi Penilaian	Parameter Pengukuran	Target Keberhasilan
1.	Fungsionalitas *Endpoint*	Memastikan *endpoint* beroperasi sesuai dengan fungsi yang telah ditetapkan	HTTP status kode	Kode 200/201 untuk *request* valid dan kode 400/401/404/500 untuk *request* invalid
2.	*Response Time*	Akomodasi waktu sistem dalam merespons *request*	*Response Time* (ms)	< 3 pada *request* umum dan < 5 detik untuk *request* klasifikasi citra

Tabel 3 Kriteria Evaluasi Pengujian *Endpoint* API (lanjutan)

No	Kriteria Evaluasi	Deskripsi Penilaian	Parameter Pengukuran	Target Keberhasilan
3.	Keamanan *Endpoint*	Validasi autentikasi dan otorisasi pengguna	*Authorization header* dan *access token*	Menghasilkan token akses apabila valid dan kode 401 (*Unauthorized*) apabila gagal serta tidak menghasilkan token
4.	Penanganan *Error*	Kejelasan dan ketepatan pesan *error* yang dihasilkan	Simulasi *request* invalid	Pesan *error* informatif dan konsisten
5.	Validasi Struktur Respons	Konsistensi format JSON dan kelengkapan atribut respons	*Body response*	Format JSON sesuai standar dan tidak berubah

Evaluasi ini menjadi dasar acuan untuk memastikan bahwa sistem *backend* telah memenuhi kriteria fungsional, keamanan, dan performa yang dibutuhkan sebelum diintegrasikan dengan sistem frontend pada tahap implementasi sistem secara menyeluruh.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Arsitektur backend yang dirancang berhasil mengintegrasikan model klasifikasi citra berbasis CNN melalui komunikasi HTTP POST antara Express.js dan layanan Cloud Run. Pengujian menunjukkan sistem mampu memproses hingga 44.400 *request* dengan *throughput* 75,07 *request*/detik, rata-rata *response time* 89–165 ms, dan P90 < 300 ms, yang menandakan performa stabil pada berbagai pola beban.
2. Implementasi *backend* berbasis Express.js terealisasi melalui 26 *endpoint* RESTful API yang mendukung autentikasi, manajemen pengguna, dan klasifikasi citra. Hasil pengujian fungsionalitas terhadap 93 skenario menunjukkan validitas 100%, dengan performa *endpoint* klasifikasi citra berada dalam batas kebutuhan non-fungsional (< 5 detik).
3. *Backend* yang dikembangkan menyediakan RESTful API yang aman dan responsif untuk mendukung aplikasi web dan mobile. Hal ini dibuktikan melalui penerapan JWT, validasi input, pembatasan ukuran *file*, serta performa stabil dengan *response time* < 200 ms, *throughput* hingga 75,07 *request*/detik, dan tanpa *error* pada seluruh skenario uji.
4. Penerapan metodologi RAD menghasilkan proses pengembangan *backend* yang terstruktur dan adaptif terhadap perubahan kebutuhan melalui dua iterasi

pengembangan yang ditunjukkan dengan penambahan fitur antar iterasi hingga seluruh kebutuhan sistem terpenuhi siap digunakan

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran pengembangan yang dapat dilakukan pada penelitian selanjutnya:

1. Pengembangan fitur analitik dan *dashboard* visualisasi data berbasis data histori klasifikasi, seperti statistik jumlah identifikasi per periode waktu, distribusi jenis lebah madu, serta pola penggunaan sistem.
2. Optimalisasi performa melalui *caching* dan *load balancing* untuk meningkatkan efisiensi dan konsistensi *response time*, sistem dapat dikembangkan dengan menerapkan mekanisme *caching* pada hasil klasifikasi tertentu serta strategi *load balancing* yang lebih terstruktur.

DAFTAR PUSTAKA

[1] S. F. Chairunissa, "Klasifikasi Jenis Lebah Madu Menggunakan Metode Convolutional Neural Network (CNN)," Skripsi, Universitas Multi Data Palembang, Indonesia, 2023.

[2] D. De Nart, et al., "Image Recognition Using Convolutional Neural Networks for Classification of Honey Bee Subspecies," *Apidologie*, vol. 53, no. 1, 2022, doi: 10.1007/s13592-022-00918-5.

[3] J. R. Valdivia, J. Gamboa-Cruzado, and P. de la Cruz Vélez de Villa, "Systematic Literature Review on Machine Learning and its Impact on APIs Deployment," *Computacion y Sistemas*, vol. 27, no. 4, pp. 1107–1124, 2023, doi: 10.13053/CyS-27-4-4371.

[4] P. De Rosa, et al., "On the Cost of Model-Serving Frameworks: An Experimental Evaluation," *Proceedings - 2024 IEEE International Conference on Cloud Engineering, IC2E 2024*, no. May, pp. 221–232, 2024, doi: 10.1109/IC2E61754.2024.00032.

[5] H. Chen and M. A. Babar, "Security for Machine Learning-based Software Systems: A Survey of Threats, Practices, and Challenges," *ACM Computing Surveys*, vol. 56, no. 6, 2024, doi: 10.1145/3638531.

[6] S. Horchidan, et al., "Crayfish: Navigating the Labyrinth of Machine Learning Inference in Stream Processing Systems," *Advances in Database Technology - EDBT*, vol. 27, no. 3, pp. 676–689, 2024, doi: 10.48786/edbt.2024.58.

[7] M. O. Nurilawati, S. L. Yasmien, and R. Pamungkas, "Rancang Bangun Aplikasi Agroteknologi " SmartFarm " dengan Model Prediktif Berbasis Deep Learning untuk Mendukung Pertanian Berkelanjutan," *Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB)*, pp. 247–255, 2025.

[8] A. F. Bahari and A. Pramudwiatmoko, "Implementation of Rapid Application Development (RAD) Method for Mobile-Based Ice Cream Ordering Application," *MALCOM: Indonesian Journal of Machine Learning and*

Computer Science, vol. 5, no. 1, pp. 283–291, 2024, doi: 10.57152/malcom.v5i1.1747.

[9] V. Jagatha, A. Khamesipour, and S. Chung, "Cross-Platform App Development: A Comparative Study of PWAs and React Native Mobile Apps," *2023 Proceedings of the ISCAP Conference*, pp. 1–10, 2023.

[10] S. Shang, et al., "Design of Cross-Platform Information Retrieval System of Library Based on Digital Twins," *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/7999091.

[11] O. M. Ahmed, "Backend as a Service Cloud Computing Integrated with Cross-platform Mobile Development Framework to Create an E-learning Application that Works in Mobile and Web with a Single Codebase," *International Conference on Communication Engineering and Computer Science*, pp. 50–57, 2022, doi: 10.24086/cocos2022/paper.511.

[12] B. Raju Cherukuri, "Building Scalable Web Applications: Best Practices for *Backend* Architecture," *International Journal of Science and Research (IJSR)*, vol. 13, no. 10, pp. 126–139, 2024, doi: 10.21275/es24928085711.

[13] M. S. Rahaman, et al., "Access Control Design Practice and Solutions in Cloud-Native Architecture: A Systematic Mapping Study," *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073413.

[14] E. Hahn, *Express in Action: Writing, Building, and Testing Node.js Applications*. USA: Manning, 2016.

[15] H. A. Goh, et al., "Front-end Deep Learning Web Apps Development and Deployment: a Review," *Applied Intelligence*, vol. 53, no. 12, pp. 15923–15945, 2023, doi: 10.1007/s10489-022-04278-6.

[16] M. Reddy, *API Design for C++*. Burlington, Elsevier, 2024.

[17] B. E. Perron, et al., "Demystifying Application Programming Interfaces (APIs): Unlocking the Power of Large Language Models and Other Web-based AI Services in Social Work Research Brian," J*ournal of the Society for Social Work and Research*, pp. 1–35, 2024.

[18] Y. Chi, *Integrating APIs with AI: Revolutionizing Modern Applications*, *ResearchGate,* pp. 1-5 , 2024.

[19] Z. M. Yusuf and R. M. Awangga, *Clean Arsitektur*. Bandung Barat, Penerbit Buku Pedia, 2023.

[20] S. Dalimunthe, E. Hasri Putra, and M. A. Fadhly Ridha, "Restful API Security Using JSON Web Token (JWT) With HMAC-Sha512 Algorithm in Session Management," *IT Journal Research and Development*, vol. 8, no. 1, pp. 81–94, 2023, doi: 10.25299/itjrd.2023.12029.

[21] I. Fawwaz, et al., "Classification of Beetle Type using the Convolutional Neural Network algorithm," *Sinkron*, vol. 7, no. 4, pp. 2340–2348, 2022, doi: 10.33395/sinkron.v7i4.11673.

[22] S. Sitio, *Penerapan Metode Rapid Application Development (Rad) Untuk Aplikasi E Learning Berbasis Web*, Purbalingga, Penerbit Cv.Eureka Media Aksara, 2023.

[23] M. F. A. Assyifa and R. Andarsyah, *Tutorial Membangun Aplikasi Notifikasi Preventive Maintenance Asset*. Kreatif, 2020.

[24] F. A. Nariswari, et al., "Sistemasi: Jurnal Sistem Informasi Rancang Bangun Aplikasi Klasifikasi Citra Rempah Famili Zingiberaceae dengan Metode Rapid Application Development Design and Development of Spice Image Classification Application of Zingiberaceae Family with Rapid Applic," *Jurnal Sistem Informasi (Sistemasi)*, vol. 13, no. 5, pp. 2540–9719, 2024.

[25] A. Fauzi dan Y. A. H. Fauzi, *TESTING DAN QA PERANGKAT LUNAK: Memahami Konsep, Metode, Teknik, dan Pendekatan dalam Pengujian*. Purbalingga, Penerbit Cv.Eureka Media Aksara, 2021.

[26] O. James, *Mastering Postman: A Comprehensive Guide to Building End-to-End APIs with Testing, Integration and Automation*. GitforGits, 2023.

[27] F. Khoiriyah, et al., *Penggunaan Cloud Services di GCP untuk Back-end API*, Banyumas, Wawasan Ilmu, 2024.

[28] P. Nagargoje, "Design and Implementation of a Real-Time CNN-Based Web Platform for Automated Skin Disease Diagnosis," *International Journal for Research in Applied Science and Engineering Technology*, vol. 13, no. 6, pp. 538–542, 2025, doi: 10.22214/ijraset.2025.72123.

[29] M. Melinda, et al., "Design and Implementation of Mobile Application for CNN-Based EEG Identification of Autism Spectrum Disorder," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 14, no. 1, pp. 57–64, 2024, doi: 10.18517/ijaseit.14.1.19676.

[30] A. Ramadhani, N. Iriadi, and R. Hidayat, "Implementasi Teknologi Rest Api Dengan Node Js Untuk Aplikasi Rekomendasi Destinasi Wisata," *Indonesian*

Journal Computer Science, vol. 4, no. 1, pp. 22–29, 2025, doi: 10.31294/t3z8vz27.

[31] A. A. Kumar and Divya TL, "Security Measures Implemented in RESTful API Development," *Open Access Research Journal of Engineering and Technology*, vol. 7, no. 1, pp. 105–112, 2024, doi: 10.53022/oarjet.2024.7.1.0042.

[32] N. Nguyen, "Development & Deployment of a Web Server as an Executable with Node.js, Express.js and Vercel/pkg," Thesis, Metropolia University of Applied Sciences, Helsinki, Finlandia, 2022.

[33] V. Azkarin, R. G. Guntara, and O. Herdiana, "Development of a REST API for Human Resource Information System for Employee Referral Management Domain Using the Express JS Framework and Node.js," *Journal of Scientific Research, Education, and Technology (JSRET)*, vol. 2, no. 3, pp. 1085–1094, 2023, doi: 10.58526/jsret.v2i3.199.

[34] A. Vu, "Real-Time Backend Architecture Using Node.js, Express and Google Cloud Platform Author Title Number of Pages Date," Thesis, Metropolia University of Applied Sciences, Helsinki, Finlandia, 2021.

[35] A. B. Semma, et al., "Cloud computing: google firebase firestore optimization analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 29, no. 3, pp. 1719–1728, 2023, doi: 10.11591/ijeecs.v29.i3.pp1719-1728.

[36] F. Cahyo and W. M. Baihaqi, "Implementasi dan Deployment Model Machine Learning Menggunakan App Engine pada Google Cloud Platform," *Journal of Informatics and Interactive Technology*, vol. 1, no. 3, pp. 197–205, 2024, doi: 10.63547/jiite.v1i3.19.

[37] O. Banias, et al., "Automated specification-based testing of REST APIs," *Sensors*, vol. 21, no. 16, 2021, doi: 10.3390/s21165375.