IMPLEMENTASI ARSITEKTUR *MICROSERVICE* PADA APLIKASI PENDETEKSI KEPITING SOKA

LAPORAN KERJA PRAKTIK

Disusun Oleh:

Gabrielle Natasha Arla Frederica

20013014

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS KATOLIK DE LA SALLE

MANADO

2023

IMPLEMENTASI ARSITEKTUR *MICROSERVICE* PADA APLIKASI PENDETEKSI KEPITING SOKA

LAPORAN KERJA PRAKTIK

Ditulis untuk Memenuhi Persyaratan Mata Kuliah Kerja Praktik

(INF2217401)

Disusun Oleh:

Gabrielle Natasha Arla Frederica

20013014

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS KATOLIK DE LA SALLE

MANADO

2023

LEMBAR PENGESAHAN

LAPORAN KERJA PRAKTIK

Judul:

IMPLEMENTASI ARSITEKTUR *MICROSERVICE* PADA APLIKASI PENDETEKSI KEPITING SOKA

Telah disetujui dan disahkan pada tanggal: 31 Januari 2024

Oleh:

Bangkit Academy

Dita Anggraini Ekawati
Cohort Manager

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama	: Gabrielle Natasha Arla Frederica
NIM	: 20013014
Tempat/Tanggal Lahir	: Tangerang, 19 Desember 2001
Fakultas/Program Studi	:Teknik/Teknik Informatika

Menyatakan bahwa laporan Kerja Praktik dan atau sistem berjudul **Implementasi Arsitektur *Microservice* Pada Aplikasi Pendeteksi Kepiting Soka** yang telah saya buat adalah benar hasil karya saya dan bukan karya tulis orang lain, baik sebagian atau seluruhnya kecuali dalam bentuk kutipan yang telah disebutkan sumbernya.

Demikian surat ini saya buat dengan sebenar-benarnya dan apabila pernyataan ini tidak benar maka saya bersedia menerima sanksi akademis sesuai dengan yang ditetapkan oleh Fakultas Teknik, berupa pembatalan Kerja Praktik dan hasilnya.

Manado, 31 Januari 2024

Yang Menyatakan,

F88AKX772004960

Gabrielle Natasha Arla Frederica

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Steven Pandelaki, S.T., M.Sc.	**Vivie Deyby Kumenap, S.T., M.C.S.**

Mengetahui,

Ketua Program Studi	Dekan Fakultas Teknik
Vivie Deyby Kumenap, S.T., M.C.S.	**Ronald A. Rachmadi, S.T., M.T.**

FORM KP - 004

FORMULIR DATA UMUM PERUSAHAAN

NAMA MAHASISWA	: Gabrielle Natasha Arla Frederica
NIM	: 20013014
NAMA PERUSAHAAN	: Yayasan Dicoding Indonesia
ALAMAT PERUSAHAAN	: Dicoding Space, Jalan Batik Kumeli No 50, Kecamatan: Cibeunying Kaler, Kelurahan: Sukaluyu, RT: 10, RW: 07, Kota Bandung, Jawa Barat, 40123
DIDIRIKAN TAHUN	: 2018
IJIN USAHA	: NIB (9120304611285)
BIDANG BISNIS	: Jasa Pendidikan Komputer (Teknologi Informasi)
JUMLAH KARYAWAN	: 95
PEMILIK	: Narenda Wicaksono & Kevin Kurniawan
DEWAN DIREKTUR	: Dimas Catur Wibowo
	: Nur Rohman
	: Ahmad Imaduddin
WAKIL PERUSAHAAN	
Tanggal	: 16 Januari 2024
Nama	: Kevin Kurniawan
Jabatan	: Pendiri Yayasan Dicoding Indonesia
(Tanda tangan dan cap perusahaan)	:

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS KATOLIK DE LA SALLE MANADO

FORM KP - 004

FORMULIR PENILAIAN KEMAJUAN KERJA PRAKTEK

A. UMUM

Nama Mahasiswa	: Gabrielle Natasha Arla Frederica
NIM Mahasiswa	: 20013014
Program Studi	: Teknik Informatika
Dosen Pembimbing Akademik	: Vivie Deyby Kumenap, S.T., M.C.S.
Topik/Rencana Bidang	: Implementasi Arsitektur *Microservice* Pada Aplikasi Pendeteksi Kepiting Soka
Pembimbing 1	: Steven Pandelaki, S.T., M.Sc.
Terhitung Mulai	: 14 Agustus 2023
Target Selesai	: 31 Januari 2024

B. KEGIATAN PELAKSANAAN KERJA PRAKTEK

No.	Tanggal	Jenis Kegiatan	Paraf Pembimbing
1.	4 Oktober 2023	Konsultasi Topik *Capstone*	
2.	4 Oktober 2023	Konsultasi Topik *Capstone*	
3.	17 Oktober 2023	Konsultasi Proposal *Capstone* Bangkit	
4.	17 Oktober 2023	Konsultasi Proposal *Capstone* Bangkit	
5.	23 Oktober 2023	Konsultasi Judul Laporan Kerja Praktik	
6.	23 Oktober 2023	Konsultasi Judul Laporan Kerja Praktik	
7.	31 Oktober 2023	Konsultasi BAB I Pendahuluan	

8.	31 Oktober 2023	Konsultasi BAB I Pendahuluan	
9.	14 November 2023	Revisi BAB I Pendahuluan	
10.	17 November 2023	Konsultasi BAB II dan BAB III	
11.	24 November 2023	Revisi BAB II dan BAB III	
12.	30 November 2023	Konsultasi BAB IV Pembahasan	
13.	7 Desember 2023	Revisi BAB IV Pembahasan	
14.	14 Desember 2023	Konsultasi BAB IV Pembahasan	
15.	4 Januari 2024	Revisi BAB IV Pembahasan	
16.	12 Januari 2024	Revisi BAB IV Pembahasan	

Manado, 31 Januari 2024

Dosen Pembimbing KP

(Steven Pandelaki, S.T., M.Sc.)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS KATOLIK DE LA SALLE MANADO

FORM KP - 005

FORMULIR PENILAIAN PELAKSANAAN KERJA PRAKTEK

Mohon diisi dan dicek seperlunya,

NAMA MAHASISWA	: Gabrielle Natasha Arla Frederica
NIM	: 20013014
NAMA PERUSAHAAN	: Yayasan Dicoding Indonesia
ALAMAT PERUSAHAAN	: Dicoding Space, Jalan Batik Kumeli No 50, Kecamatan: Cibeunying Kaler, Kelurahan: Sukaluyu, RT: 10, RW: 07, Kota Bandung, Jawa Barat, 40123
TGL KERJA PRAKTEK	: 14 Agustus 2023 s.d 31 Desember 2023
TOPIK YANG DIBAHAS	: Cloud Computing Learning Path

Nilai Sikap	=	50	60	70	80	(90)	100
Kerajinan	=	50	60	70	80	(90)	100
Prestasi	=	50	60	70	80	(90)	100

KOMENTAR/SARAN

Mahasiswa telah mengikuti kegiatan dengan baik.

NILAI RATA-RATA	: 90
TANGGAL	: 16 Januari 2024
NAMA PENILAI	: Deti Anggraini Ekawati
JABATAN	: Cohort Manager

(Tanda tangan dan cap perusahaan) : 

KATA PENGANTAR

Pujian dan rasa syukur kepada Tuhan Yang Maha Esa atas berkat dan tuntunan-Nya sehingga penulis dapat menyelesaikan Laporan Kerja Praktik di Yayasan Dicoding Indonesia dalam program Magang dan Studi Independen Bersertifikat (MSIB) dengan judul Implementasi Arsitektur *Microservice* Pada Aplikasi Pendeteksi Kepiting Soka, dengan baik.

Laporan Kerja Praktik ini disusun sebagai bagian dari upaya untuk memenuhi salah satu persyaratan akademik program studi Teknik Informatika untuk memberikan hasil dan dokumentasi selama mengikuti program MSIB di Yayasan Dicoding Indonesia.

Selama mengikuti program MSIB hingga tahap penyusunan Laporan Kerja Praktik ini tidak terlepas dari bantuan dan dukungan berbagai pihak yang turut serta berkontribusi, baik secara langsung maupun tidak langsung. Terlebih khusus kepada:

1. Prof. Dr. Johanis Ohoitimur, selaku Rektor dari Universitas Katolik De La Salle Manado.
2. Bapak Ronald Albert Rachmadi, S.T., M.T., selaku Dekan Fakultas Teknik
3. Ibu Vivie Deyby Kumenap, S.T., M.C.S., selaku Ketua Program Studi Teknik Informatika, Dosen Pembimbing Akademik penulis, dan Dosen Pembimbing II yang selalu membimbing dan mendukung penulis selama masa perkuliahan, serta selaku *Supervisor* selama penulis mengikuti program MSIB di Yayasan Dicoding Indonesia.
4. Bapak Michael George Sumampouw, S.T., M.T., selaku Koordinator Instansi Universitas Katolik De La Salle Manado selama penulis mengikuti program MSIB di Yayasan Dicoding Indonesia.
5. Bapak Steven Pandelaki, S.T., M.Sc., selaku Dosen Pembimbing I yang telah membantu, membimbing dan mendukung penulis dalam menyelesaikan penyusunan Laporan Kerja Praktik dengan baik.
6. Pihak Tim Bangkit dan Kampus Merdeka yang sudah memberikan kesempatan, pengalaman, serta pengetahuan selama penulis mengikuti program MSIB.

7. Ellyas I. Sinaga, selaku penasihat dan juga pembimbing yang selalu membantu dan memberikan arahan kepada penulis selama mengikuti program MSIB.
8. Orang tua, keluarga serta pasangan Gabriel F. Tumewu dan keluarga yang selalu mendukung dan membantu penulis dalam belajar dan mencapai setiap titik tujuan dalam hidup penulis.
9. Annastassya Christina Karundeng, selaku sahabat baik yang selalu memberikan dorongan dan semangat untuk penulis.
10. Teman-teman Angkatan 2020 program studi Teknik Informatika yang telah berjuang bersama penulis dan memberikan semangat persaudaraan satu sama lain.

Penulis ingin menyampaikan terima kasih dan rasa syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa serta seluruh pihak terkait. Laporan Kerja Praktik ini tentunya tidak terlepas dari kesalahan dan kekurangan sehingga kritik dan saran yang membangun sangat diharapkan untuk perbaikan serta pengembangan di masa depan.

Manado, 31 Januari 2024

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Kepiting bakau dan kepiting soka merupakan salah satu komoditas ekspor unggulan Indonesia yang diminati secara luas. Kepiting bakau, yang umumnya hidup di hutan bakau, memiliki nilai konsumsi tinggi karena kemampuannya untuk mengalami pergantian cangkang (*molting*), menjadikan seluruh bagian tubuhnya lunak dan dapat dikonsumsi secara utuh[1]. Keunggulan gizi kepiting, yang rendah lemak, tinggi protein, dan kaya vitamin serta mineral, membuatnya menjadi pilihan yang diminati oleh pasar lokal dan global[2].

Dengan produksi kepiting di Indonesia mencapai lebih dari 50.000 ton per tahun, kepiting soka, yang memiliki cangkang lunak, menjadi komoditas ekspor dengan harga yang relatif tinggi. Pada tahun 2023, harga kepiting soka mencapai Rp.235.000/kg dengan berat rata-rata ±200 gr/ekor[3]. Meskipun peluang ini menjanjikan, pengusaha budidaya kepiting soka sering menghadapi kendala, seperti lamanya periode pemeliharaan dan waktu *molting* yang tidak seragam. Hal ini mengakibatkan perlunya pemantauan harian untuk mencegah penurunan produksi[3].

Di era teknologi saat ini, pengembang teknologi berlomba-lomba menawarkan solusi menggunakan *Internet of Things*, dan *Machine Learning*. Tantangan muncul dalam mengintegrasikan teknologi-teknologi ini ke dalam solusi, terutama ketika menggunakan pendekatan monolitik yang menggabungkan semua fungsi dalam satu entitas tunggal[4]. Pendekatan ini seringkali menyulitkan skalabilitas, pemeliharaan, dan inovasi. Sebagai solusi, pendekatan *microservices* muncul, memecah aplikasi menjadi komponen-komponen kecil yang dapat beroperasi mandiri, memungkinkan skalabilitas yang lebih baik, pemeliharaan yang lebih mudah, dan inovasi yang lebih fleksibel. Pemanfaatan layanan *cloud computing* juga menjadi kunci dalam mengimplementasikan dan mengelola *microservices* secara efisien.

Dalam menghadapi kendala budidaya kepiting soka, aplikasi pendeteksi kepiting soka yang menggunakan teknologi *machine learning* dapat

diimplementasikan dengan efektif melalui layanan *cloud computing*. Layanan ini memainkan peran kunci dalam mengintegrasikan model *machine learning* dengan aplikasi Android, menyediakan berbagai manfaat termasuk peningkatan skalabilitas, kemudahan pemrosesan data, pengembangan model *machine learning*, penyimpanan data dan keamanan data.

Penerapan layanan *cloud computing microservice* menjadi solusi yang relevan untuk mengatasi tantangan dalam mendeteksi dan memonitor produksi kepiting soka. Dengan demikian, dapat diharapkan bahwa integrasi teknologi ini akan mempercepat proses pendeteksian, meningkatkan efisiensi, dan mengoptimalkan hasil pendeteksian kepiting soka. Keseluruhan solusi ini diharapkan dapat memberikan dampak positif terhadap pengembangan aplikasi pendeteksi kepiting soka, memberikan kontribusi pada konservasi keanekaragaman hayati, dan mendukung upaya pemeliharaan keberlanjutan ekosistem *mangrove* di Indonesia.

1.2. Rumusan Masalah

Bagaimana mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka?

1.3. Tujuan Kerja Praktik

Mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka?

1.4. Manfaat Kerja Praktik

1. Bagi Pengembang Aplikasi

a. Meningkatkan efisiensi fungsi aplikasi pendeteksi kepiting soka menggunakan layanan *cloud*.

b. Meningkatkan kualitas perangkat lunak dengan menerapkan layanan *cloud computing* menggunakan arsitektur *microservices*.

2. Bagi Penulis

a. Memperoleh pengetahuan dan pengalaman dalam melakukan implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka.

b. Memperoleh pengetahuan dan pengalaman dalam belajar *Cloud Computing*.

1.5. Batasan Masalah

Selama proses penelitian ini dibuat beberapa batasan dengan maksud untuk memfokuskan penelitian ini pada tujuan utamanya berdasarkan tujuan kerja praktik.

1. Membahas seputar layanan *cloud computing* berupa Firebase *Authentication*, *Cloud Firestore, Cloud Storage Bucket*, dan TensorFlow *Lite.*
2. Fokus pada implementasi arsitektur *microservice* berupa *Cloud* Firestore, *Cloud Storage* Bucket, dan TensorFlow *Lite* pada aplikasi pendeteksi kepiting soka.
3. Pengujian model *machine learning* TensorFlow *Lite* yang diimplementasikan pada Flask API diuji menggunakan Postman.

1.6. Sistematika Penulisan

Berikut ini merupakan rincian dari sistematika penulisan dalam laporan kerja praktik ini yang bertujuan untuk membuat laporan kerja praktik ini lebih terstruktur dengan memisahkan setiap modul dalam beberapa bab, sebagai berikut:

1. BAB I PENDAHULUAN

Pada bab ini membahas mengenai latar belakang masalah, tujuan kerja praktik, manfaat kerja praktik, batasan masalah, serta rincian sistematika penulisan laporan kerja praktik ini.

2. BAB II DATA UMUM PERUSAHAAN

Pada bab ini membahas mengenai sejarah singkat perusahaan, lingkup pekerjaan perusahaan, serta lingkup pekerjaan yang dilakukan selama melakukan kerja praktik.

3. BAB III LANDASAN TEORI

Pada bab ini memberikan beberapa teori yang dijadikan sebagai landasan atau dasar dari penelitian yang dilakukan dalam laporan kerja praktik ini, mencakup teori-teori secara umum mengenai topik yang diangkat, mengenai metodologi dan

algoritma yang digunakan dalam penelitian ini, serta metode pengumpulan data yang dilakukan.

4. BAB IV PEMBAHASAN

Pada bab ini menjelaskan mengenai pengumpulan dan pengolahan data yang lebih detail dan terfokus pada tujuan penelitian, analisis pemecahan masalah yang dilakukan, serta membahas mengenai proses perancangan hingga pembuatan algoritma dan kode program.

5. BAB V KESIMPULAN DAN SARAN

Pada bab ini diberikan kesimpulan dari penelitian yang telah dilakukan untuk mengetahui ringkasan hasil penelitian serta memberikan saran untuk pengembangan lanjutan dari hasil penelitian ini

BAB II
DATA UMUM PERUSAHAAN

2.1. Sejarah Singkat Perusahaan

Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi (Kemendikbudristek) Indonesia mencerminkan evolusi kebijakan pemerintah dalam mengelola sektor pendidikan, kebudayaan, riset, dan teknologi. Pada awalnya, sebelum restrukturisasi pada tahun 2020, Kementerian ini dikenal sebagai Kementerian Pendidikan dan Kebudayaan (Kemendikbud), yang didirikan pada tahun 1945[5]. Seiring berjalannya waktu, pentingnya pengembangan riset dan teknologi dalam mendukung pertumbuhan bangsa semakin terasa, sehingga pada tahun 2020, pemerintah memutuskan untuk menggabungkan Kemendikbud dengan Kementerian Riset dan Teknologi (Ristek)[6].

Gambar 2.1 Logo Kemendikbudristek[7]

Pergabungan ini bertujuan untuk menciptakan sinergi antara pendidikan, kebudayaan, riset, dan teknologi guna mempercepat kemajuan Indonesia di berbagai bidang. Dengan restrukturisasi ini, terbentuklah Kemendikbudristek yang menjadi entitas tunggal yang mengawasi dan mengelola sektor-sektor tersebut. Sejarah Kemendikbudristek mencerminkan komitmen pemerintah Indonesia dalam meningkatkan kualitas sumber daya manusia dan mendorong inovasi untuk mencapai pembangunan yang berkelanjutan[6].

2.1.1. Kampus Merdeka

Program Merdeka Belajar – Kampus Merdeka (MBKM) diumumkan secara resmi oleh Menteri Pendidikan dan Kebudayaan pada awal tahun 2020 melalui sejumlah peraturan yang ditetapkan. Program ini diinisiasi sebagai solusi untuk mengatasi berbagai tantangan yang dihadapi oleh Perguruan Tinggi dalam menyiapkan lulusan yang mampu beradaptasi dengan dinamika perkembangan zaman, kemajuan ilmu pengetahuan, teknologi, serta memenuhi kebutuhan dari dunia usaha dan industri. MBKM lahir sebagai respons terhadap tuntutan akan peningkatan kualitas lulusan yang tidak hanya memiliki keahlian di bidang studi utama mereka, tetapi juga kompetensi tambahan yang relevan dengan kebutuhan pasar kerja[8].

Program MBKM menawarkan delapan program yang memberikan hak istimewa kepada mahasiswa untuk mengikuti pembelajaran di luar ruang lingkup bidang studi mereka yang utama. Keistimewaan ini diwujudkan dengan memberikan batas waktu maksimum selama tiga semester, setara dengan akumulasi 60 sks (Satuan Kredit Semester). Mahasiswa dapat memanfaatkan waktu tersebut untuk mengikuti mata kuliah, workshop, atau kegiatan pembelajaran lainnya yang mungkin tidak termasuk dalam kurikulum utama mereka. Pendekatan ini bertujuan untuk memberikan ruang lebih besar kepada mahasiswa agar dapat mengembangkan potensi, minat, dan keterampilan di luar ranah akademis utama mereka.

Gambar 2.2 Logo Kampus Merdeka[9]

MBKM tidak hanya menekankan aspek akademis semata, tetapi juga merangkul dinamika masyarakat dan kebutuhan industri. Dengan demikian,

mahasiswa diharapkan dapat menjadi lulusan yang tidak hanya berkualifikasi akademis tinggi tetapi juga siap dan mampu berkontribusi secara aktif dalam dunia kerja. Adanya program ini menjadi langkah inovatif dalam mendukung transformasi pendidikan tinggi di Indonesia, menciptakan lulusan yang tidak hanya cerdas secara akademis, tetapi juga memiliki keterampilan, pengetahuan, dan karakter yang relevan dengan tuntutan dunia kerja yang terus berkembang. Seiring berjalannya waktu, Program MBKM diharapkan dapat memberikan dampak positif dalam mencetak generasi penerus yang tangguh dan mampu menghadapi perubahan global.

2.1.2. Bangkit *Academy*

Bangkit *Academy* merupakan sebuah program yang dirancang sejak tahun 2020 untuk mempersiapkan karir dengan tujuan menghasilkan bakat dan keahlian yang profesional dalam bidang teknologi dan perusahaan *startup*. Program ini didukung secara penuh oleh perusahaan Google, GoTo, dan Traveloka untuk memberikan dukungan berupa sarana pembelajaran *online* yang terstruktur dan relevan dengan kebutuhan industri terkini[10].

Gambar 2.3 Logo Bangkit *Academy*[10]

2.2. Lingkup Pekerjaan Perusahaan

Bangkit *Academy* bergerak dalam sektor industri pendidikan, terutama dengan penekanan pada pembelajaran elektronik (*e-learning*). Dengan tekad untuk menjadi pelopor dalam pengembangan sumber daya manusia, Bangkit *Academy* bertujuan memberikan kontribusi besar dalam peningkatan kemampuan digital individu. Dalam rentang waktu antara tahun 2020 hingga 2023, program-program yang ditawarkan oleh akademi ini telah membentuk tiga jalur

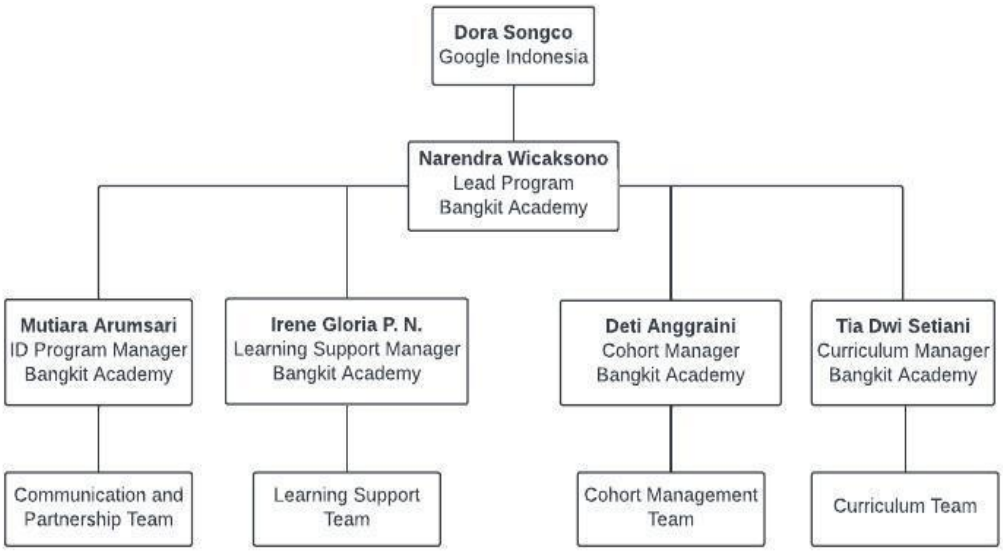
pembelajaran utama yang mencakup Android, *Machine Learning*, dan *Cloud Computing*[11]. Setiap *learning path* menyediakan materi yang terstruktur dari tingkat dasar hingga tingkat profesional. Kurikulum yang digunakan juga berdasarkan pada standar industri global yang dikembangkan bersama perusahaan dan pemilik teknologi dunia sehingga bisa menyesuaikan dengan kebutuhan industri terkini. Kerja sama antara program Bangkit *Academy* dengan Kampus Merdeka, memberikan kesempatan yang lebih luas bagi mahasiswa untuk belajar dengan materi yang relevan dan lebih mendalam karena berfokus pada perkembangan industri terkini.

Dalam rangka mencapai kualitas pembelajaran yang optimal, Bangkit *Academy* memastikan bahwa kurikulum yang digunakan didasarkan pada standar industri global. Kerja sama erat dengan perusahaan-perusahaan dan pemilik teknologi terkemuka di seluruh dunia menjadi landasan utama penyusunan kurikulum ini. Hal ini bertujuan untuk memastikan bahwa materi pembelajaran selalu relevan dan mampu menyesuaikan diri dengan dinamika industri teknologi yang terus berkembang.

Sebagai wujud upaya Bangkit *Academy* untuk memberikan pengalaman belajar yang holistik, akademi ini menjalin kemitraan dengan Kampus Merdeka. Kolaborasi ini memberikan mahasiswa akses yang lebih luas untuk memperdalam pemahaman mereka, dengan fokus pada materi-materi yang tidak hanya relevan tetapi juga berpusat pada perkembangan terbaru dalam industri. Dengan demikian, Bangkit *Academy* dan Kampus Merdeka menciptakan sinergi yang unik, membuka pintu bagi para mahasiswa untuk memperluas wawasan dan keterampilan mereka sesuai dengan tuntutan industri yang terus berubah.

2.2.1. Struktur Organisasi

Struktur organisasi di Bangkit *Academy* dirancang dengan cermat untuk menciptakan lingkungan kerja yang kooperatif dan efektif. Dalam kerangka ini, masing-masing anggota organisasi memiliki peran dan tanggung jawab tertentu yang mendukung pencapaian tujuan akademi. Berikut adalah gambaran lebih rinci tentang struktur organisasi dan pembagian tingkat kepentingan tugas dan tanggung jawab di Bangkit *Academy*.



Gambar 2.4 Struktur Organisasi Bangkit *Academy*[11]

Dengan adanya struktur organisasi yang terstruktur dengan baik dan pembagian tugas yang jelas, Bangkit *Academy* dapat mencapai efisiensi operasional dan memberikan pengalaman pendidikan yang berkualitas tinggi bagi para pesertanya. Sinergi antara berbagai divisi ini menciptakan lingkungan yang mendukung pertumbuhan dan inovasi di bidang teknologi.

2.3. Lingkup Pekerjaan yang dilakukan

Selama menjalani kerja praktik, penulis terlibat dalam serangkaian kegiatan yang mencakup pemahaman mendalam tentang *Cloud Computing* melalui partisipasi dalam kursus pembelajaran daring. Keaktifan dalam mengikuti kelas praktisi memberikan wawasan praktis dalam penggunaan teknologi *cloud*. Selain itu, kelas pengembangan *softskill* dan Bahasa Inggris juga menjadi bagian integral dari upaya pengembangan diri selama periode kerja praktik.

Rutinitas mingguan penulis melibatkan kehadiran konsultasi setiap hari Selasa, di mana penulis berkesempatan untuk berdiskusi dan berinteraksi dengan mentor terkait progres dan tantangan yang dihadapi selama proses pembelajaran. Kegiatan ini tidak hanya memperkaya pengetahuan tetapi juga memberikan pandangan praktis tentang penerapan konsep-konsep yang telah dipelajari dalam situasi dunia nyata.

Sebagai bagian dari tantangan yang dihadapi selama kerja praktik, penulis bertanggung jawab untuk merancang suatu produk berbasis teknologi (*Product Based*) untuk proyek *Capstone*. Proses ini melibatkan analisis data yang cermat untuk memahami kebutuhan dan tujuan produk. Penulis juga terlibat dalam merancang dan mengimplementasikan arsitektur *microservice* sebagai *back-end* untuk memastikan kehandalan dan fungsionalitas produk yang dihasilkan.

Selain fokus pada aspek teknis, penulis juga berperan dalam membuat laporan dokumentasi yang komprehensif terkait hasil *Capstone Project* dan pencapaian selama periode kerja praktik. Dokumentasi ini tidak hanya bertujuan untuk merekam setiap langkah dan pengambilan keputusan, tetapi juga untuk memberikan panduan yang berguna dan transparan bagi pihak terkait.

Melalui serangkaian kegiatan ini, penulis berhasil mengembangkan keterampilan teknis dan *softskill*, memperdalam pemahaman tentang *Cloud Computing*, dan melibatkan diri secara aktif dalam pengembangan proyek berbasis teknologi. Ini tidak hanya menjadi tantangan yang membangun, tetapi juga pengalaman yang sangat berharga dalam mempersiapkan diri untuk tantangan di dunia kerja yang semakin kompleks.

BAB III
LANDASAN TEORI

3.1. Teori Pendukung

Dalam konteks penelitian ini, sejumlah teori pendukung menjadi landasan utama dalam pengembangan dan implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka. Konsep arsitektur *microservices* menjadi teori kunci yang mendasari pendekatan pengembangan perangkat lunak. Arsitektur *microservices* membawa paradigma di mana sebuah aplikasi dikonstruksi sebagai kumpulan layanan kecil, independen, dan terpisah. Setiap layanan berfokus pada fungsi atau fitur tertentu, dan mereka dapat berkomunikasi satu sama lain melalui antarmuka yang ditentukan. Pendekatan ini menawarkan sejumlah keunggulan, seperti skalabilitas yang lebih baik, fleksibilitas dalam pengembangan dan pembaruan, serta kemudahan dalam penyembunyian detail implementasi.

Berikut ini merupakan beberapa teori pendukung yang digunakan sebagai dasar dalam penelitian untuk mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka.

3.1.1. Kepiting Soka

Kepiting soka merupakan sebutan untuk kepiting cangkang lunak yang diperoleh pada saat kepiting bakau sedang mengganti cangkang kulitnya. Kepiting bakau itu sendiri merupakan salah satu jenis kepiting yang hidup di perairan hutan bakau (*mangrove*). Berikut ini merupakan klasifikasi kepiting soka berdasarkan taksonominya[12].

Kepiting soka diklasifikasikan dalam kerajaan Animalia, yang berarti ia termasuk dalam kelompok makhluk hidup multiseluler dan eukariotik. Dalam filum Arthropoda, kepiting soka merupakan bagian dari kelompok hewan yang memiliki kerangka luar, anggota tubuh bersendi, dan seringkali memiliki eksoskeleton yang keras. Kelas Krustasea menunjukkan bahwa kepiting soka adalah krustasea, yaitu kelompok hewan air yang mencakup udang, lobster, dan kepiting.

Gambar 3.1 Kepiting Soka[13]

Sub kelas Malakostraka menunjukkan bahwa kepiting soka termasuk dalam kelompok krustasea yang memiliki tubuh yang lebih lunak dibandingkan dengan kelompok lainnya dalam kelas Krustasea. Dengan ordo Decapoda, kepiting soka termasuk dalam kelompok hewan yang memiliki sepuluh kaki yang termodifikasi, yang mencakup kepiting dan lobster. Infratakson Brachyura menunjukkan bahwa kepiting soka termasuk dalam kelompok kepiting sesungguhnya, yang ditandai dengan ciri-ciri seperti karapas yang lebih lebar dan perut yang lebih pendek. Superfamili Portunoidea menunjukkan hubungan keluarga yang lebih dekat dengan keluarga-keluarga kepiting lainnya.

Famili Portunidae menandakan bahwa kepiting soka termasuk dalam keluarga yang mencakup berbagai jenis kepiting, termasuk yang hidup di perairan tropis dan subtropis. Genus Scylla menunjukkan bahwa kepiting soka termasuk dalam kelompok genus yang mencakup beberapa jenis kepiting air tawar dan air laut. Spesies Scylla serrata menunjukkan jenis kepiting soka yang memiliki ciri khas tertentu dalam spesiesnya. Dengan demikian, taksonomi ini memberikan informasi rinci tentang penempatan kepiting soka dalam hierarki ilmiah dan karakteristiknya yang khusus.

Siklus hidup yang dimiliki oleh kepiting ini dimulai saat telur kepiting menetas, maka akan menjadi larva dan selama masa pertumbuhannya akan terus mengalami pergantian cangkang kulit (*molting*) sebanyak 5 kali mulai dari fase *zoea* 1 hingga fase *zoea* 5 selama ±21 hari. Setelahnya kepiting akan masuk dalam

fase *megalopa* dimana capit sudah mulai terbentuk dan akan mengalami *molting* sebanyak 2 kali selama ±1 minggu. Kemudian kepiting akan mulai bertumbuh menjadi kepiting muda dan akan mengalami *molting* sebanyak ±15 kali untuk menjadi kepiting dewasa. Dalam setiap proses *molting* kepiting biasanya akan bertumbuh sebesar sepertiga kali dari ukuran sebelumnya. Setelah kepiting dewasa, maka akan melewati proses kawin dan menetaskan telur[1].

3.1.2. *Vertical Crab House*

Vertical crab house atau yang sering disebut sebagai apartemen kepiting menawarkan sebuah inovasi revolusioner dalam dunia akuakultur, khususnya dalam budidaya kepiting soka di daratan. Sebagai alternatif yang berbeda dari budidaya tradisional di tambak, konsep apartemen kepiting ini menciptakan suatu lingkungan yang unik dan terkontrol untuk meningkatkan kualitas hidup dan pertumbuhan kepiting. Berbeda dengan tambak yang terbuka, apartemen kepiting didesain dengan kegelapan dan tertutup, menciptakan suasana yang lebih kondusif untuk menjaga kestabilan lingkungan budidaya[14].

Gambar 3.2 *Vertical Crab House*[15]

Satu elemen kunci dari apartemen kepiting adalah pendekatan sistem resirkulasi dan filtrasi yang digunakan. Sistem ini bertujuan untuk menyaring kotoran dan menjaga kualitas air tetap optimal. Dengan memanfaatkan teknologi ini, air dalam sistem dapat dijaga kebersihannya dan dapat digunakan kembali, menciptakan suatu siklus tertutup yang efisien. Hal ini tidak hanya mendukung

efisiensi penggunaan air, tetapi juga membantu mengurangi dampak lingkungan yang dapat dihasilkan oleh pembuangan air yang tidak terkendali.

Selain itu, desain apartemen kepiting juga memperhitungkan aspek perilaku alami kepiting soka. Setiap unit apartemen didesain untuk memisahkan setiap kepiting, mencegah sifat kanibalisme yang mungkin terjadi di antara mereka. Pemisahan ini tidak hanya berdampak positif pada kesejahteraan kepiting, tetapi juga memastikan bahwa masing-masing kepiting dapat tumbuh dengan optimal tanpa adanya gangguan atau persaingan yang berlebihan.

Dengan memanfaatkan prinsip-prinsip ini, apartemen kepiting menjadi solusi inovatif untuk meningkatkan efisiensi dan keberlanjutan dalam budidaya kepiting soka. Pendekatan ini menciptakan suatu sistem yang lebih terkendali, ramah lingkungan, dan memperhatikan aspek kesejahteraan hewan. Dengan demikian, apartemen kepiting tidak hanya menjadi solusi praktis untuk petani kepiting, tetapi juga mencerminkan upaya dalam menghadirkan teknologi yang mendukung keberlanjutan sektor akuakultur secara keseluruhan.

3.1.3. Teknologi Pengembangan Perangkat Lunak

Pada bagian ini akan dijelaskan mengenai teori-teori pendukung terkait teknologi pengembangan perangkat lunak yang akan digunakan dalam mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka.

3.1.3.1. Arsitektur *Microservices*

Microservices merupakan pendekatan dalam pengembangan perangkat lunak dimana aplikasi dibangun sebagai kumpulan layanan yang kecil, independen, dan terpisah. Setiap layanan dalam arsitektur *microservices* fokus pada fungsi atau fitur tertentu dan dapat berkomunikasi dengan layanan lainnya melalui antarmuka yang ditentukan[4]. Beberapa keunggulan dari *Microservices*:

1. Skalabilitas, layanan dapat diubah dan diperbarui secara independen, memungkinkan skalabilitas yang lebih baik. Jika suatu layanan memerlukan peningkatan kapasitas, hanya layanan tersebut yang perlu diubah.

2. Fleksibilitas, mudah untuk memperbarui atau menambahkan fitur tanpa mengganggu layanan lainnya. Setiap layanan dapat dikembangkan, diuji, dan diimplementasikan secara independen.
3. Penyembunyian Detail Implementasi, layanan mengekspos *endpoint* sebagai API, menyembunyikan semua detail implementasi internal seperti logika, arsitektur, dan teknologi.
4. Pengembangan yang cepat, tim pengembangan dapat bekerja secara independen pada setiap layanan, memungkinkan pengembangan yang lebih cepat dan pembaruan yang lebih sering.
5. Ketersediaan dan Ketahanan, jika satu layanan mengalami kegagalan, layanan lainnya masih dapat beroperasi, meningkatkan ketersediaan dan ketahanan keseluruhan sistem.
6. Teknologi yang berbeda, setiap layanan dapat menggunakan teknologi yang paling sesuai untuk tugasnya, tidak terbatas pada satu set teknologi tertentu.
7. Mudah dikelola, memungkinkan untuk pemantauan dan manajemen yang lebih baik, karena setiap layanan dapat dikelola secara terpisah.

Meskipun terdapat sejumlah keunggulan, pendekatan *microservices* juga memiliki tantangan, seperti kompleksitas manajemen, koordinasi antar layanan, dan kebutuhan untuk komunikasi jaringan yang baik antar layanan.

3.1.3.2. Arsitektur Monolitik

Arsitektur monolitik adalah pendekatan sederhana dan alami dalam membangun aplikasi di mana seluruh logika berjalan dalam satu proses. Pendekatan ini memiliki keuntungan kesederhanaan dalam pengujian, implementasi, *debugging*, dan pemantauan. Namun, seiring dengan pertumbuhan aplikasi, masalah mulai muncul, seperti kesulitan mengelola kompleksitas kode, penurunan kinerja, dan kesulitan dalam membuat perubahan. Jumlah pengembang yang meningkat juga dapat menyebabkan ketidakseimbangan dalam pemanfaatan tenaga kerja dan penurunan produktivitas[4].

Semua komponen dan fungsi aplikasi terkait dan tergantung satu sama lain. Dalam sebuah monolit, jika satu bagian mengalami masalah atau perlu diperbarui, seluruh aplikasi perlu diperbarui dan di *deploy* ulang[4].

3.1.3.3. *Cloud Computing*

Cloud computing adalah model komputasi yang memungkinkan akses yang mudah dan *on-demand* ke sejumlah sumber daya komputasi, seperti server, penyimpanan data, jaringan, basis data, perangkat lunak, analitika, dan kecerdasan buatan, melalui internet. Sebagai pengganti dari penyimpanan dan pemrosesan data secara lokal pada komputer pribadi atau server fisik, *cloud computing* memanfaatkan infrastruktur yang tersebar secara global. Ada beberapa karakteristik utama dari *cloud computing*, yaitu[16]:

1. Akses *On-Demand*

   Pengguna dapat dengan mudah mengakses dan menggunakan sumber daya komputasi sesuai kebutuhan, tanpa perlu memiliki atau mengelola infrastruktur secara fisik.

2. Elastisitas dan Skalabilitas

   Cloud computing memungkinkan untuk penyesuaian otomatis terhadap kebutuhan pengguna. Sumber daya dapat ditingkatkan atau dikurangi sesuai dengan permintaan, sehingga memberikan fleksibilitas dan efisiensi yang tinggi.

3. *Self-Service* dan Pengelolaan Otomatis

   Pengguna dapat mengelola dan mengontrol sumber daya komputasi melalui antarmuka mandiri. Selain itu, banyak aspek dari infrastruktur dapat diotomatisasi untuk meningkatkan efisiensi.

4. Akses melalui Internet

   Sumber daya komputasi dalam cloud diakses melalui jaringan internet. Hal ini memungkinkan pengguna untuk bekerja dari mana saja dengan koneksi internet.

5. Pembayaran Berbasis Penggunaan (*Pay-as-You-Go*)

   Pengguna hanya membayar untuk sumber daya yang mereka gunakan, seperti layaknya biaya utilitas. Ini dapat membantu mengoptimalkan biaya dan mencegah pemborosan.

6. Pemisahan Fisik Infrastruktur

   Infrastruktur komputasi secara fisik terletak di pusat data yang dapat tersebar di berbagai lokasi geografis. Pengguna tidak perlu tahu secara spesifik di mana data atau aplikasi mereka disimpan.

7. Keamanan

Penyedia layanan *cloud* bertanggung jawab untuk menyediakan tingkat keamanan yang tinggi, termasuk perlindungan data, enkripsi, dan kontrol akses.

Beberapa model layanan *cloud* utama termasuk *Infrastructure as a Service* (IaaS) menyediakan infrastruktur dasar seperti server dan penyimpanan, *Platform as a Service* (PaaS) menyediakan platform pengembangan, dan *Software as a Service* (SaaS) menyediakan aplikasi perangkat lunak yang dapat diakses langsung melalui web[16].

3.1.3.4. *Google Cloud Platform*

Google Cloud Platform (GCP) adalah kumpulan layanan *cloud* yang disediakan oleh Google untuk membantu pengembang dan organisasi menyusun, menyimpan, dan mengelola aplikasi dan data mereka di lingkungan *cloud*. GCP menawarkan berbagai layanan, termasuk komputasi, penyimpanan data, analisis, kecerdasan buatan, *machine learning*, dan layanan lainnya yang dapat digunakan untuk membangun dan menjalankan aplikasi skala besar. *Google Cloud Platform* memungkinkan pengembang untuk membangun aplikasi dengan skala global, dengan keamanan tinggi, dan dapat diakses dengan mudah. GCP bersaing dengan *platform cloud* lainnya seperti Amazon Web Services (AWS) dan Microsoft Azure[17].

3.1.3.5. *Cloud Storage Bucket*

Cloud storage bucket adalah istilah yang umumnya digunakan dalam layanan komputasi awan (*cloud computing*), khususnya pada penyimpanan objek. Secara umum, sebuah *bucket* dalam konteks *cloud storage* merujuk kepada wadah atau tempat penyimpanan utama untuk menyimpan objek digital, seperti *file*, gambar, *video*, atau data lainnya. Setiap *bucket* memiliki nama yang unik dalam ruang nama penyimpanan, dan dapat diakses dan dikelola melalui antarmuka pengguna atau melalui API yang disediakan oleh penyedia layanan *cloud*[18].

Keuntungan utama dari menggunakan *cloud storage bucket* termasuk kemudahan dalam menyimpan, mengakses, dan berbagi data secara terdistribusi, skalabilitas yang baik untuk menangani data yang besar, dan ketersediaan data yang tinggi melalui infrastruktur yang terdistribusi di berbagai lokasi

geografis[18]. Misalnya, dalam layanan *Google Cloud Storage*, dapat digunakan untuk membuat *bucket* yang dapat menyimpan objek-objek digital. Nama *bucket* tersebut harus unik di dalam proyek *Google Cloud*, dan pengguna dapat mengatur izin akses untuk mengontrol siapa yang dapat mengakses data di dalam *bucket* tersebut.

3.1.3.6. Firebase

Firebase adalah *platform* pengembangan aplikasi berbasis *cloud* yang dimiliki oleh Google. Firebase menyediakan berbagai layanan dan alat yang dapat membantu pengembang membangun, mengelola, dan meningkatkan aplikasi dengan lebih mudah. Firebase menawarkan solusi lengkap yang mencakup aspek-aspek seperti pengembangan aplikasi, basis data *real-time*, autentikasi pengguna, penyimpanan file, penggunaan server tanpa server (*serverless*), dan lainnya. Berikut adalah beberapa fitur utama Firebase, yaitu[19]:

1. *Real-time Database*

   Firebase menyediakan basis data NoSQL *real-time* yang memungkinkan pengembang menyinkronkan dan menyimpan data di seluruh klien secara instan. Ini sangat berguna untuk aplikasi yang memerlukan pembaruan data secara langsung.

2. Autentikasi

   Firebase menyediakan layanan auntentikasi pengguna yang mudah digunakan. Ini memungkinkan pengembang untuk mengimplementasikan autentikasi dengan berbagai metode, seperti *e-mail* atau *password*, autentikasi sosial media (Google, Facebook, Twitter), dan lainnya.

3. *Cloud Firestore*

   Firestore adalah layanan basis data yang menyediakan skema dokumen dan koleksi. Ini memberikan fleksibilitas dalam menyimpan dan mengelola data aplikasi dengan kemampuan yang tinggi.

4. *Storage*

   Firebase *Storage* memungkinkan penyimpanan dan pengelolaan file, seperti gambar atau rekaman, di *cloud*. Ini berguna untuk menyimpan dan menyajikan konten media kepada pengguna aplikasi.

5. *Cloud Functions*

Firebase *Cloud Functions* memungkinkan pengembang menulis dan menjalankan kode server tanpa harus mengelola server fisik. Ini memungkinkan pengembang untuk merespons peristiwa di aplikasi mereka dengan mudah.

6. *Hosting*

Firebase *Hosting* menyediakan infrastruktur *hosting* untuk aplikasi web statis. Ini memudahkan pengembang untuk menyajikan situs web atau aplikasi web dengan cepat dan mudah.

7. *Cloud Messaging*

Firebase *Cloud Messaging* (FCM) memungkinkan pengembang mengirim *push notification* ke perangkat pengguna mereka. Hal ini memungkinkan interaksi *real-time* dengan pengguna.

8. *Performance Monitoring dan Analytics*

Firebase menyediakan alat untuk memantau performa aplikasi dan menganalisis perilaku pengguna. Ini membantu pengembang memahami bagaimana aplikasi digunakan dan mengidentifikasi area-area yang dapat dioptimalkan. Firebase merupakan pilihan yang populer di kalangan pengembang aplikasi karena menyederhanakan banyak aspek pengembangan, memberikan skalabilitas, dan memungkinkan fokus pada fungsionalitas utama aplikasi[19].

3.1.3.7. TensorFlow

TensorFlow merupakan sebuah perangkat lunak sumber terbuka (*open source*) yang dikembangkan oleh tim Google Brain. Ini adalah *platform machine learning* yang dirancang untuk memudahkan pengembangan dan pelatihan model *machine learning*. TensorFlow menyediakan kerangka kerja yang komprehensif untuk membangun dan melatih berbagai jenis model *machine learning*, termasuk *neural networks* untuk tugas-tugas seperti klasifikasi gambar, pemrosesan bahasa alami, dan banyak lagi[20].

TensorFlow menggunakan representasi data dalam bentuk tensor, yang merupakan struktur data multi dimensi dengan derajat tinggi. Ini memungkinkan pengguna untuk menyusun dan melatih model *machine learning* secara efisien. Selain itu, TensorFlow memiliki antarmuka yang mudah digunakan dan

mendukung berbagai bahasa pemrograman seperti Python, Java, C++, dan lainnya. TensorFlow juga memiliki alat visualisasi yang kuat untuk memahami dan menganalisis performa model, serta mendukung pelatihan model di berbagai platform, termasuk CPU, GPU, dan TPU (*Tensor Processing Unit*). Kehadiran TensorFlow telah menjadi faktor kunci dalam percepatan dan kemajuan dalam pengembangan aplikasi *machine learning* dan *deep learning*[20].

3.1.3.8. TensorFlow *Lite*

TensorFlow *Lite* adalah versi ringan (*lightweight*) dari TensorFlow, dikembangkan khusus untuk perangkat seluler dan perangkat dengan sumber daya terbatas seperti mikrokontroler (perangkat *edge* atau perangkat *Internet of Things*). Tujuannya adalah untuk memungkinkan penerapan model *machine learning* di lingkungan yang memiliki keterbatasan daya komputasi dan sumber daya[21].

TensorFlow *Lite* memungkinkan penanaman (*embedding*) model *machine learning* langsung ke dalam aplikasi seluler atau perangkat *edge*, memungkinkan pengambilan keputusan lokal tanpa koneksi internet. Ini sangat berguna untuk aplikasi cerdas (*smart*) yang memerlukan deteksi objek, klasifikasi gambar, pemrosesan bahasa alami, dan tugas *machine learning* lainnya secara langsung di perangkat tanpa mengandalkan server eksternal[21]. Dengan menyediakan format model yang dioptimalkan untuk sumber daya terbatas, TensorFlow *Lite* memungkinkan pengembang untuk membuat aplikasi cerdas yang lebih efisien dan responsif di perangkat seluler dan *edge*, sehingga dapat digunakan di berbagai skenario, termasuk di lingkungan IoT.

3.1.3.9. Python

Python adalah bahasa pemrograman tingkat tinggi yang mendukung berbagai paradigma pemrograman, seperti pemrograman prosedural, berorientasi objek, dan fungsional. Dikembangkan oleh Guido van Rossum dan dirilis pada tahun 1991, Python memprioritaskan keterbacaan kode dengan menggunakan indentasi untuk menandai blok kode. Bahasa ini bersifat interpretatif dan dinamis,

memungkinkan pengembangan kode tanpa memerlukan proses kompilasi terlebih dahulu[22].

Python memiliki ekosistem modul dan pustaka yang kaya, mendukung pengembangan dengan lebih mudah dengan memanfaatkan fungsionalitas yang telah ada. Fleksibilitas Python tercermin dalam kemampuannya untuk berjalan di berbagai *platform* tanpa perlu modifikasi signifikan, menjadikannya portabel. Bahasa ini juga membanggakan dukungan yang kuat untuk pengelolaan memori otomatis dan sistem tipe data dinamis[22].

Tabel 3.1 Contoh Kode Program Python

Kode Program
print("Hello, World!”)

Dengan komunitas yang aktif dan beragam, Python digunakan secara luas dalam berbagai bidang, termasuk pengembangan web dengan Django dan Flask, ilmu data dengan NumPy dan Pandas, kecerdasan buatan dengan *TensorFlow* dan PyTorch, serta berbagai proyek perangkat lunak. Filosofi Python, dikenal sebagai "*The Zen of Python*," merangkum prinsip-prinsip desain yang menekankan nilai-nilai seperti keterbacaan, kesederhanaan, dan ekspresivitas dalam penulisan kode[22].

3.1.3.10. Flask

Flask adalah sebuah *framework* web mikro yang ditulis dalam bahasa pemrograman Python. Dikembangkan oleh Armin Ronacher, Flask dirancang untuk menjadi ringan, fleksibel, dan mudah digunakan, memberikan dasar yang sederhana untuk membangun aplikasi web. Meskipun Flask dianggap sebagai *framework* web mikro, ia menyediakan alat-alat yang cukup untuk membangun aplikasi web dengan fitur lengkap. Salah satu fitur utama dari Flask adalah filosofi "Werkzeug dan Jinja," yang menyediakan dasar untuk manajemen rute, *template*, dan fitur web lainnya[23].

Flask memberikan kebebasan kepada pengembang untuk memilih alat dan pustaka pihak ketiga yang akan digunakan, memungkinkan pengembang untuk membangun aplikasi web sesuai dengan kebutuhan dan preferensi mereka. Flask

tidak memerlukan struktur proyek yang ketat, dan ini membuatnya sangat cocok untuk pemula yang ingin memahami dasar-dasar pengembangan web[23].

Dengan menggunakan Flask, pengembang dapat dengan cepat membuat API (*Application Programming Interface*), aplikasi *mobile*, web atau layanan web mikro dengan menjalankan server pengembangan *built-in* atau dengan menyebarkannya pada server produksi. Flask juga mendukung ekstensi yang memperluas fungsionalitas bawaan, seperti manajemen sesi, autentikasi pengguna, dan lainnya[23].

Tabel 3.2 Contoh Kode Program Flask

Kode Program

from flask import Flask
 	
app = Flask(__name__)
 	
@app.route('/')
def hello_world():
  return 'Hello, World!'
 	
if __name__ == '__main__':
  app.run(debug=True)

Pada Tabel 3.2, aplikasi Flask diinisialisasi, dan satu rute ("/") didefinisikan untuk mengembalikan pesan "Hello, World!" saat halaman ditemukan. Flask kemudian dijalankan menggunakan server pengembangan bawaan.

3.1.3.11. *Application Programming Interface*

Application Programming Interface (API) adalah seperangkat aturan dan protokol yang memungkinkan dua *software* atau aplikasi untuk berkomunikasi satu sama lain. API menyediakan cara untuk satu aplikasi menggunakan layanan atau fungsi yang disediakan oleh aplikasi lain tanpa perlu mengetahui detail internal implementasinya. API bertindak sebagai perantara yang memungkinkan interaksi antara dua sistem yang berbeda[24].

API dapat berbentuk berbagai jenis, termasuk API *web*, API *database*, API perangkat keras, dan lain sebagainya. API menyediakan antarmuka yang terstandarisasi untuk mengakses fungsionalitas tertentu atau mendapatkan data dari suatu aplikasi atau layanan. Penggunaan API memungkinkan pengembang

membuat aplikasi yang dapat berintegrasi dengan aplikasi atau layanan lain tanpa perlu memahami seluruh implementasi internal. API juga memfasilitasi pengembangan cepat dan integrasi sistem yang lebih mudah dalam pengembangan perangkat lunak[24].

3.1.3.12. Postman

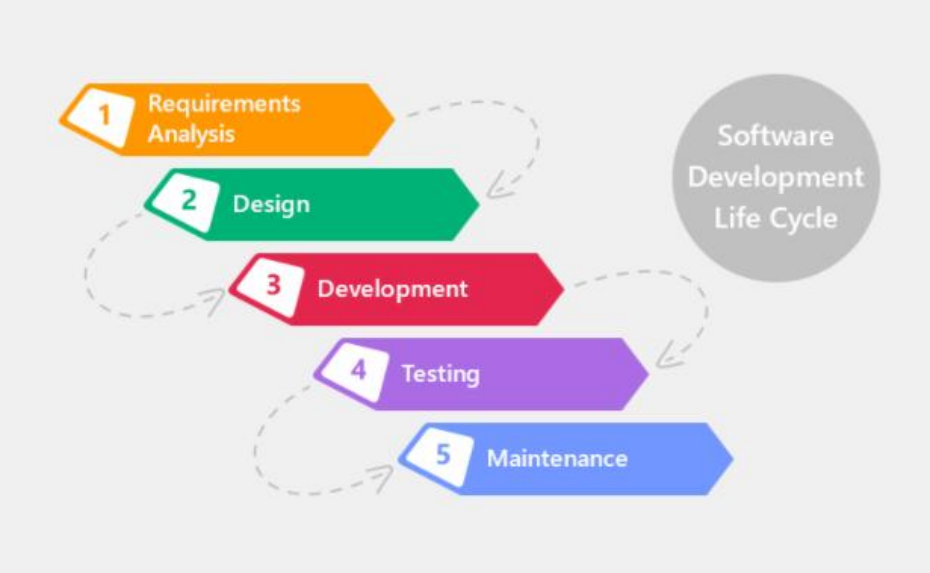
Postman adalah sebuah *platform* kolaboratif yang memainkan peran kunci dalam pengujian dan pengembangan API. Dengan antarmuka pengguna grafis yang intuitif, Postman memudahkan para pengembang untuk membuat, mengirim, dan menerima permintaan HTTP dengan berbagai metode seperti *GET*, *POST*, *PUT*, dan *DELETE*. Kemampuan untuk mengatur permintaan API ke dalam koleksi mempermudah manajemen dan pencarian, memberikan struktur yang terorganisir[25].

Salah satu fitur unggulan Postman adalah kemampuannya untuk berbagi dan berkolaborasi. Pengguna dapat menyimpan koleksi dan permintaan API, lalu membagikannya dengan anggota tim, mendukung kerja sama efisien dalam pengembangan proyek. Postman juga mendukung pengelolaan variabel, memungkinkan pengguna menyimpan dan menggunakan nilai variabel secara dinamis dalam permintaan API, yang sangat berguna untuk mengatasi perubahan lingkungan atau kebutuhan yang kompleks[25].

Postman tidak hanya sebatas alat pengujian, namun juga menyediakan lingkungan uji yang kuat. Pengguna dapat menulis dan menjalankan skrip pengujian API serta mengotomatiskan pengujian untuk memverifikasi fungsionalitas API secara otomatis. Selain itu, Postman dapat digunakan untuk membuat dokumentasi API yang komprehensif, termasuk deskripsi permintaan, respons, dan skenario penggunaan. Dokumentasi ini dapat dibagikan dan diakses oleh tim pengembang, memudahkan pemahaman dan penggunaan API. Dengan fitur-fitur tersebut, Postman menjadi alat yang sangat berharga dalam siklus pengembangan dan pengujian API, mendukung efisiensi dan kolaborasi tim pengembang[25].

3.2. Metodologi Pengembangan Sistem

Metodologi yang digunakan untuk pengembangan sistem dalam mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka berbasis Android adalah metodologi *Waterfall*. Metodologi *Waterfall* merupakan metode dengan pendekatan pengembangan perangkat lunak yang terstruktur dengan mengikuti 5 tahapan metodologi pengembangan perangkat lunak[26].



Gambar 3.3 Tahapan Metodologi *Waterfall*[27]

3.2.1. Tahapan Metodologi *Waterfall*

Pada bagian ini akan dijelaskan mengenai 5 tahapan yang ada dalam metodologi *Waterfall*, yaitu sebagai berikut[28].

1. *Requirements Analysis*

Pada tahap ini akan dilakukan pengumpulan data untuk memahami kebutuhan target pengguna, memahami kebutuhan sistem yang akan dibuat, serta melakukan analisis data yang telah dikumpulkan. Tahap ini bertujuan untuk mendapatkan informasi serta persyaratan yang diperlukan sesuai dengan kebutuhan target pengguna berdasarkan rumusan masalah yang diangkat dalam penelitian ini.

2. *Design*

Pada tahap ini akan dibuat perancangan alur sistem, perancangan struktur perangkat lunak, serta membuat desain tampilan aplikasi Android untuk aplikasi

yang akan dibuat. Tahap ini bertujuan untuk memastikan bahwa rancangan dibuat telah sesuai dengan persyaratan yang ada pada tahap sebelumnya.

3. *Development*

Pada tahap ini proses pengembangan sistem dengan mengimplementasikan arssitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka sudah mulai dibuat berdasarkan tahap desain sebelumnya. Tahap ini bertujuan untuk merealisasikan rancangan yang telah dibuat.

4. *Testing*

Pada tahap ini akan dilakukan uji coba setelah selesai mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka berbasis Android. Tahap ini bertujuan untuk menguji seluruh alur kerja sistem yang telah dibangun, kemudian memastikan bahwa aplikasi dapat berjalan dengan baik dan telah memenuhi persyaratan serta mencapai tujuan dari penelitian ini.

5. *Maintenance*

Pada tahap ini aplikasi yang telah melewati tahap uji coba, selanjutnya akan dipelihara untuk tetap menjaga fungsionalitas aplikasi berjalan sesuai dengan alur kerja yang telah dibuat. Dalam tahap ini juga dapat dilakukan peningkatan aplikasi lebih lanjut dengan melakukan *update* pada fungsi-fungsi atau menambahkan fitur-fitur lain sebagai pengembangan dari aplikasi yang telah dibuat sebelumnya. Namun dalam penelitian ini tidak akan dilanjutkan hingga tahap ini.

3.2.2. Kakas Pemodelan

Penelitian ini mendesain metodologi implementasi dengan memanfaatkan pemodelan data terstruktur, seperti *Flowchart* dan *Data Flow Diagram* (DFD), sebagai alat utama untuk menggambarkan secara rinci alur kerja sistem yang akan dikembangkan. Pendekatan ini bertujuan untuk memberikan gambaran yang komprehensif tentang bagaimana sistem bekerja dan berinteraksi dengan data dalam lingkup aplikasi pendeteksi kepiting soka.

Flowchart digunakan sebagai instrumen visual yang efektif untuk mewakili langkah-langkah proses secara berurutan. Dengan menggunakan *Flowchart*, penelitian ini dapat memvisualisasikan proses-proses utama dalam Aplikasi

Pendeteksi Kepiting Soka, membantu pemahaman dan analisis yang lebih baik terkait alur kerja aplikasi.

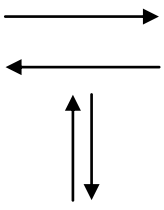
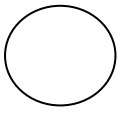
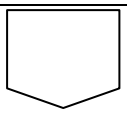
Selain itu, *Data Flow Diagram* (DFD) digunakan sebagai alat untuk memodelkan alur data dalam sistem. DFD memungkinkan representasi visual dari pergerakan data dari satu proses ke proses lainnya, menciptakan gambaran yang jelas tentang bagaimana informasi mengalir melalui aplikasi.

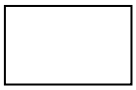
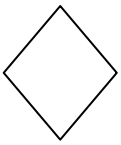
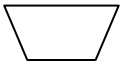
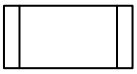
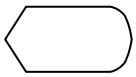
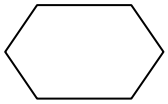
Kombinasi antara Flowchart dan DFD memberikan kerangka kerja yang kokoh untuk merinci langkah-langkah operasional dan arus data yang diperlukan oleh sistem. Pemilihan pendekatan pemodelan ini didasarkan pada kebutuhan untuk memahami secara menyeluruh struktur dan fungsionalitas aplikasi. Dengan demikian, diharapkan bahwa pemodelan data terstruktur ini akan menjadi panduan yang efektif dalam proses pengembangan dan implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka.

3.2.2.1. *Flowchart*

Flowchart adalah representasi grafis atau diagram yang menggambarkan alur atau urutan langkah-langkah dalam suatu proses atau sistem. *Flowchart* digunakan untuk memvisualisasikan secara sistematis serangkaian tindakan atau keputusan yang diambil dalam suatu algoritma atau prosedur[29].

Tabel 3.3 Simbol *Flowchart* [29]

Simbol	Nama Simbol	Keterangan
	*Flow*	Simbol ini digunakan untuk menghubungkan antara simbol satu dengan lainnya. Simbol ini juga dikenal sebagai *connecting line*.
	*On-Page Reference*	Simbol untuk keluar-masuk atau penyambungan proses dalam lembar kerja yang sama.
	*Off-Page Reference*	Simbol untuk keluar-masuk atau penyambungan proses dalam lembar kerja yang berbeda.

Simbol	Nama Simbol	Keterangan
	*Terminator*	Simbol yang menyatakan awal atau akhir suatu program.
	*Process*	Simbol yang menyatakan suatu proses yang dilakukan oleh komputer.
	*Decision*	Simbol yang menunjukkan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban yaitu ya dan tidak.
	*Input/Output*	Simbol yang menyatakan proses *input* atau *output* tanpa bergantung pada peralatan.
	*Manual Operation*	Simbol yang menyatakan suatu proses yang tidak dilakukan oleh komputer.
	*Document*	Simbol yang menyatakan bahwa *input* berasal dari dokumen dalam bentuk fisik, atau *output* yang perlu dicetak.
	*Predefine Process*	Simbol untuk pelaksanaan suatu bagian (sub-program) atau prosedur.
	*Display*	Simbol yang menyatakan peralatan *output* yang digunakan.
	*Preparation*	Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal.

3.2.2.1. *Data Flow Diagram*

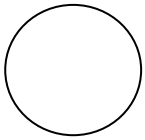
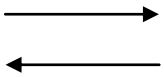
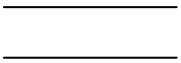
Data Flow Diagram (DFD) adalah bentuk diagram pemodelan data yang sangat berguna dalam mewakili aliran data dalam suatu sistem informasi. Dengan menggunakan DFD, kita dapat dengan jelas menggambarkan bagaimana data bergerak dari satu proses ke proses lainnya dalam sistem. DFD membantu dalam menyajikan visualisasi yang terstruktur tentang bagaimana data disimpan, diproses, atau diteruskan di dalam sistem[29].

DFD terdiri dari berbagai simbol yang mewakili proses, entitas, aliran data, dan penyimpanan data. Proses dalam DFD mewakili kegiatan atau operasi yang

memanipulasi data, sedangkan entitas mewakili sumber atau tujuan data dalam sistem. Aliran data menunjukkan arus data dari satu proses ke proses lainnya, dan penyimpanan data menggambarkan tempat penyimpanan data dalam sistem[29].

Melalui DFD, kita dapat dengan jelas melihat bagaimana entitas-entitas yang berbeda, seperti manusia, sistem, atau objek, berinteraksi satu sama lain dalam sistem. DFD membantu dalam mengidentifikasi aliran data utama dalam sistem, memahami hubungan antara komponen-komponen dalam sistem, dan menyoroti titik-titik potensial untuk perbaikan atau optimalisasi[29].

Tabel 3.4 Simbol *Data Flow Diagram* [29]

Simbol	Nama Simbol	Keterangan
	*Entity*	Mewakili sumber atau tujuan data dalam sistem. Entitas bisa berupa manusia, organisasi, objek fisik, atau sistem eksternal lainnya yang berpartisipasi dalam aliran data.
	*Process*	Mewakili aktivitas atau tindakan yang mengubah atau menghasilkan data. Proses dapat mencakup pemrosesan data, perhitungan, pengambilan keputusan, dan sebagainya.
	*Data Flow*	Mewakili aliran data atau informasi antar proses, penyimpanan data, dan entitas.
	*Data Store*	Mewakili tempat penyimpanan data, seperti basis data atau file, yang digunakan oleh proses dalam sistem.

DFD menggambarkan hubungan dan interaksi antara berbagai komponen dalam suatu sistem informasi secara tingkat tinggi. Diagram ini membantu dalam memahami alur data, mengidentifikasi proses-proses yang terlibat, dan memberikan gambaran keseluruhan tentang bagaimana sistem tersebut beroperasi[29].

3.3. Prosedur Pengumpulan Data

Dalam kerangka penelitian ini, data primer menjadi elemen kunci yang diperoleh secara langsung di lokasi Laboratorium Ilmu Kelautan IPB IFMOS Ancol, Kec. Pademangan, Jakarta Utara. Proses pengambilan sampel melibatkan gambar kepiting bakau dan kepiting soka, yang nantinya akan membentuk *dataset* untuk analisis lebih lanjut.

Selain data primer, penelitian ini juga mengandalkan data sekunder sebagai elemen pendukung. Data sekunder mencakup informasi yang diperoleh secara tidak langsung, seperti jurnal, artikel, dan penelitian terkait lainnya yang telah ada sebelumnya. Pengumpulan data sekunder dilakukan melalui riset yang teliti terhadap sumber-sumber literatur yang relevan dengan fokus penelitian. Pendekatan ini memastikan bahwa penelitian dapat memanfaatkan pengetahuan yang telah ada, mengintegrasikan temuan-temuan sebelumnya ke dalam kerangka kerja penelitian saat ini.

Dengan menggunakan data primer dan sekunder, penelitian ini dapat menggabungkan dimensi pengalaman langsung dan konteks literatur dalam menganalisis serta mengembangkan aplikasi pendeteksi kepiting soka. Pendekatan ini diharapkan dapat memberikan landasan yang kuat untuk menghasilkan temuan-temuan yang komprehensif dan aplikatif dalam konteks budidaya kepiting.

BAB IV
PEMBAHASAN

4.1. *Requirements Analysis*

Pada bagian ini akan dijelaskan proses pengumpulan dan pengolahan data yang ditujukan untuk memahami kebutuhan perangkat lunak yang akan dibuat. Data-data tersebut akan dianalisis menjadi suatu informasi untuk kebutuhan implementasi pada tahap selanjutnya dalam metodologi *waterfall*.

4.1.1. Pengumpulan Data

Dalam prosesnya, pembuatan aplikasi pendeteksi kepiting soka ini dilakukan oleh 3 kelompok, yaitu kelompok *Machine Learning*, *Cloud Computing*, dan *Mobile Development* yang telah memiliki tugas dan tanggung jawab masing-masing. Dalam hal ini penulis bertanggung jawab dalam kelompok *cloud computing* sehingga data-data yang akan dikumpulkan merupakan data sekunder yang berkaitan dengan implementasi arsitektur *microservice* pada aplikasi pendeteksi kepiting soka.

Arsitektur *microservice* adalah pendekatan dalam pengembangan perangkat lunak di mana aplikasi dibangun sebagai kumpulan layanan kecil yang beroperasi secara mandiri. Setiap layanan mikro (*microservice*) memiliki tugas spesifik dan dapat dikembangkan, diimplementasikan, dan dikelola secara independen.

Dalam konteks Firebase *Authentication*, *Cloud Firestore*, *Cloud Storage Bucket* di *Google Cloud Platform* (GCP), dan *TensorFlow Lite*, dapat diperhatikan bagaimana aspek-aspek ini dapat diintegrasikan ke dalam arsitektur *microservice*.

Firebase *Authentication Microservice*, layanan ini bertanggung jawab untuk menangani proses autentikasi pengguna. Layanan ini menyediakan berbagai metode autentikasi seperti menggunakan *e-mail*, kata sandi, dan lain sebagainya. Keterkaitannya dengan *microservice* lain berfungsi sebagai layanan otonom untuk mengelola dan memverifikasi identitas pengguna aplikasi. Penggunaan layanan ini memberikan keuntungan dengan memisahkan fungsi autentikasi dari layanan lain untuk meningkatkan skalabilitas dan memungkinkan pengembangan secara mandiri.

Cloud Firestore Microservice, layanan ini menangani penyimpanan dan manajemen data berbasis dokumen di *cloud*. *Cloud Firestore* adalah *database* NoSQL yang menyimpan data dalam format koleksi dan dokumen. Keterkaitannya dengan *microservice* lain dapat terhubung dengan layanan lain untuk menyimpan dan mengambil data terkait pengguna atau hasil klasifikasi dari model *machine learning*. Penggunaan layanan ini memberikan keuntungan karena dapat melakukan manajemen data secara terpisah untuk meningkatkan skalabilitas dan memfasilitasi pengembangan secaran independen.

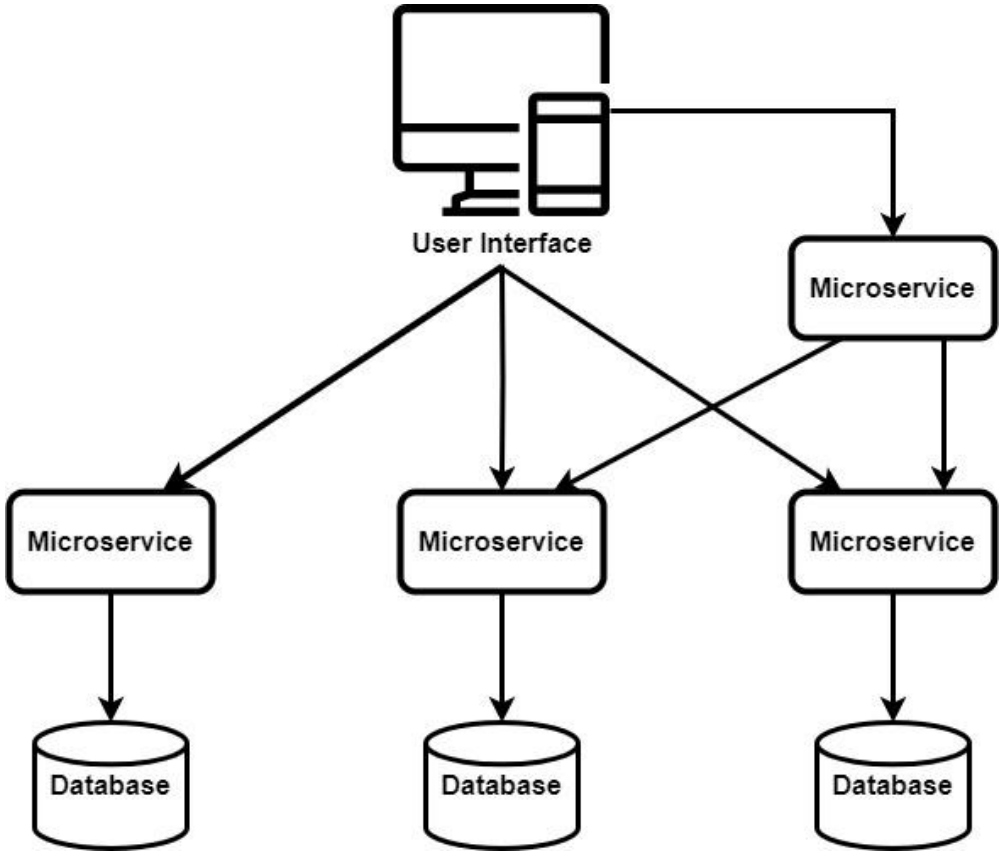
Cloud Storage Bucket, layanan ini menyediakan penyimpanan objek skala besar yang dapat diakses dari seluruh dunia. Layanan ini sangat cocok untuk menyimpan data seperti gambar atau *file* hasil pengolahan model *machine learning* yang memungkinkan memiliki ukuran *file* yang besar. Penggunaan layanan ini juga memberikan keuntungan karena menyediakan penyimpanan terdistribusi yang dapat diakses dari berbagai layanan untuk meningkatkan efisiensi dan skalabilitas.

TensorFlow Lite Microservice, layanan ini berfokus pada pemrosesan model *machine learning* di tingkat lokal (*on- device*) dengan TensorFlow *Lite*. Layanan ini juga dapat digunakan untuk melakukan klasifikasi gambar, seperti mendeteksi kepiting soka. Penggunaan TensorFlow *Lite* sebagai layanan *microservice* dapat membantu untuk mengakses data dari *Cloud Storage Bucket* dan *Cloud Firestore* untuk mendapatkakn masukkan dan mengembalikan hasil klasifikasi. Keuntungannya adalah memberikan kemampuan klasifikasi lokal yang cepat dan efisien, serta memungkinkan pengembangan model yang lebih terdistribusi.

Layanan-layanan ini terhubung melalui API yang didefinisikan dengan baik. Sebagai contoh, Firebase *Authentication* menyediakan token autentikasi yang dapat digunakan untuk mengamankan akses ke *Cloud Firestore*. Hasil klasifikasi *TensorFlow Lite* dapat disimpan di *Cloud Firestore* dan gambar yang dianalisis dapat disimpan di *Cloud Storage Bucket*.

Keuntungan penggunaan arsitektur *microservice* dalam aplikasi pendeteksi kepiting soka yaitu setiap layanannya dapat ditingkatkan untuk menanggapi permintaan yang lebih besar, dapat melakukan pemeliharaan dan pengembangan secara independen tanpa mempengaruhi layanan lain dan fleksibilitas

pengembangan layanan yang membuat pengembang dapat bekerja secara terpisah pada layanan sesuai keahlian. Skema untuk arsitektur *microservices* dapat dilihat pada Gambar 4.1 di bawah ini.



Gambar 4.1 Skema Arsitektur *Microservices*[30]

Dengan menerapkan arsitektur *microservice* dan mengintegrasikan layanan Firebase *Authentication*, *Cloud Firestore*, *Cloud Storage Bucket*, dan *TensorFlow Lite*, dapat membangun solusi yang *scaleable*, mudah dikelola, dan efisien dalam mendukung aplikasi pendeteksi kepiting soka.

4.1.2. Analisis Data dan Pemecahan Masalah

Berdasarkan data yang telah dikumpulkan, arsitektur *microservices* memberikan solusi yang efektif dalam menghadapi tantangan pengembangan aplikasi pendeteksi kepiting soka. Kelebihan arsitektur ini mencakup skalabilitas yang tinggi, pemeliharaan yang mandiri, dan fleksibilitas pengembangan. Setiap layanan mikro, seperti Firebase *Authentication*, *Cloud Firestore*, *Cloud Storage Bucket* dan *TensorFlow Lite*, memiliki tugas spesifik dan dapat dikembangkan

secara independen. Firebase *Authentication* digunakan sebagai layanan otonom untuk menangani proses autentikasi pengguna dengan metode yang aman. *Cloud Firestore*, sebagai layanan manajemen data, menyediakan penyimpanan berbasis dokumen di cloud. Begitu juga dengan *Cloud Storage Bucket*, yang memberikan penyimpanan objek skala besar yang dapat diakses dari seluruh dunia. *TensorFlow Lite*, sebagai layanan pemrosesan model *machine learning*, fokus pada klasifikasi gambar dengan pengolahan lokal.

Pengintegrasian antar layanan dilakukan melalui API yang didefinisikan dengan baik, memastikan efisiensi dan interaksi yang lancar. Sebagai contoh, token autentikasi dari Firebase *Authentication* digunakan untuk mengamankan akses ke *Cloud Firestore*. Hasil klasifikasi *TensorFlow Lite* dapat dengan mudah disimpan di *Cloud Firestore*, sementara gambar-gambar yang dianalisis disimpan di *Cloud Storage Bucket.*

Arsitektur ini memberikan solusi untuk manajemen identitas pengguna, penyimpanan dan pemrosesan data yang efisien, serta pemrosesan model *machine learning* yang cepat. Dengan memecah aplikasi menjadi layanan-layanan kecil, pengembang dapat bekerja secara terpisah pada setiap layanan, meningkatkan kecepatan pengembangan dan pemeliharaan. Melalui implementasi arsitektur *microservices* dan integrasi layanan seperti Firebase *Authentication*, *Cloud Firestore*, *Cloud Storage Bucket*, dan *TensorFlow Lite*, solusi ini diharapkan dapat memberikan aplikasi pendeteksi kepiting soka yang *scaleable*, mudah dikelola, dan efisien.

4.1.3. Spesifikasi Persyaratan Perangkat Lunak

Pada bagian ini akan diberikan daftar spesifikasi persyaratan perangkat lunak yang diperlukan untuk mengimplementasikan arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka.

4.1.4.1. Persyaratan Fungsional

Berikut ini merupakan daftar spesifikasi persyaratan perangkat lunak secara fungsional, yaitu:

a. Manajemen akses pengguna menggunakan Firebase *Authentication*.

b. Sistem dapat mengirim token autentikasi kepada pengguna untuk melakukan *sign up* dan *sign in*.
c. Sistem dapat berintegrasi dengan model *machine learning* menggunakan model *TensorFlow Lite* untuk mendeteksi kepiting soka.
d. Sistem dapat menyimpan data hasil deteksi ke *Cloud Firestore* dan menyimpan gambar pada *Cloud Storage Bucket*.

4.1.4.2. Persyaratan Non-Fungsional

Berikut ini merupakan daftar spesifikasi persyaratan perangkat lunak secara non-fungsional, yaitu:

a. Sistem dapat memberikan respons cepat dari setiap permintaan *client* terhadap *microservices.*
b. Sistem dapat menjaga pengelolaan biaya layanan *cloud* tetap efisien.
c. Pengguna dapat menerima token autentikasi dari Firebase *Authentication* sebagai bentuk keamanan pengguna.

4.1.4. Ruang Lingkup Proyek

Berikut ini merupakan bagian-bagian yang menjadi ruang lingkup dari penelitian ini, yaitu:

1. Penelitian ini dilakukan sebagai ide usulan untuk *Product Capstone Project* selama mengikuti Kerja Praktik dalam program Bangkit.
2. Penelitian ini menggunakan tema atau studi kasus *Food Accessibility, Agribusiness, and Food Security*.
3. Melakukan analisis terhadap implementasi arsitektur *microservice* pada aplikasi pendeteksi kepiting soka.
4. Mengumpulkan data serta melakukan riset pada penelitian-penelitian terkait topik serupa yang sudah ada sebelumnya.
5. Membuat perancangan arsitektur *microservices*.
6. Mengimplementasikan hasil rancangan yang telah dibuat.
7. Melakukan uji coba sistem yang telah dibuat.

4.2. *Design*

Pada bagian ini akan dijelaskan rancangan sistem menggunakan *Flowchart* dan *Data Flow Diagram* (DFD) serta membuat desain arsitektur *microservices* yang akan dibuat sehingga dapat digunakan sebagai dasar perencanaan dalam membangun sistem *back-end*.

4.2.1. Pemodelan Sistem

Berikut ini merupakan pemodelan sistem yang akan dibuat dengan *flowchart* untuk merepresentasikan alur sistem *back-end* dan *data flow diagram* untuk merepresentasikan alur data dari sistem yang akan dibuat.

4.2.1.1. *Flowchart*

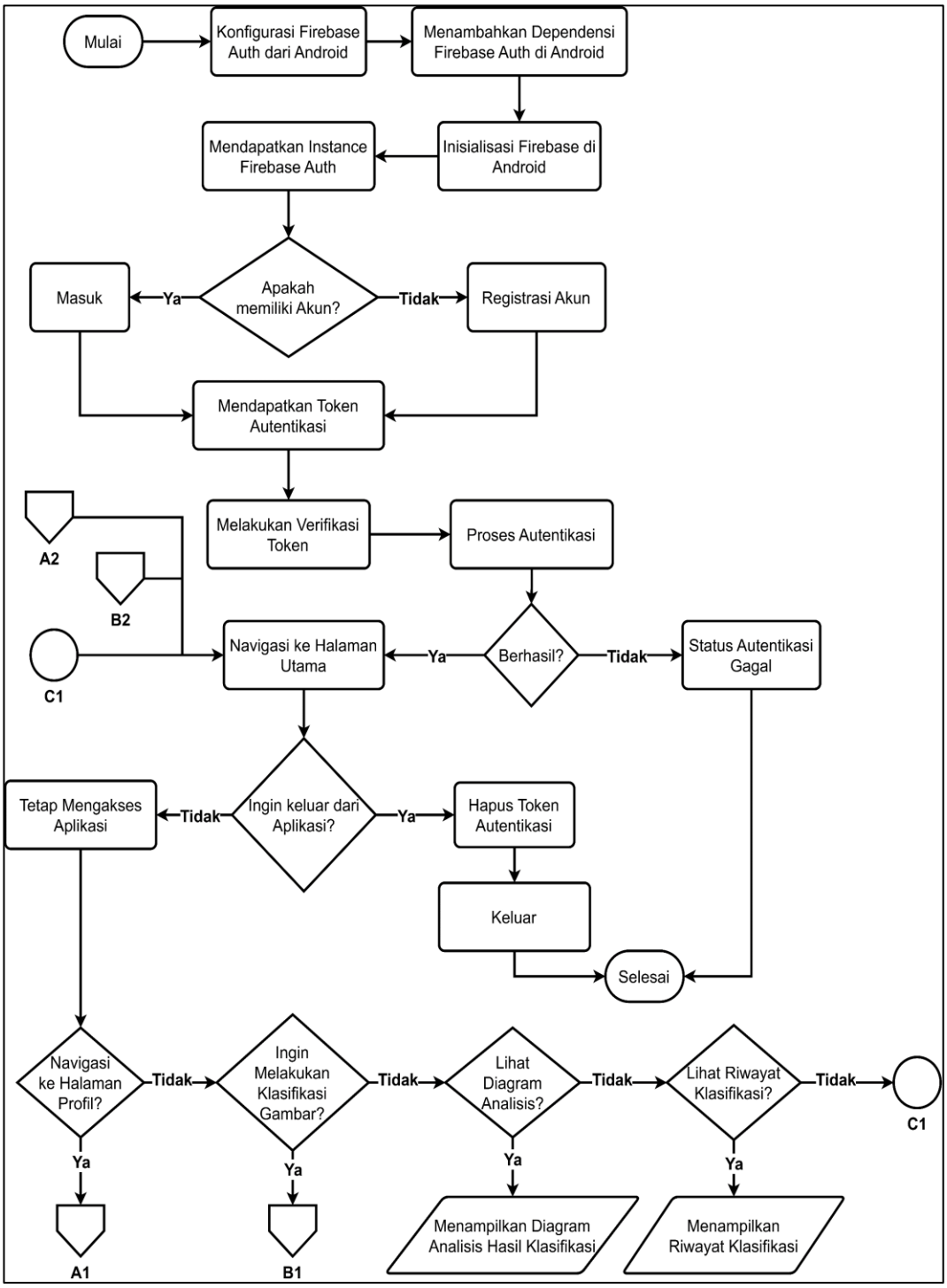
Pada Gambar 4.2 merepresentasikan gambaran desain *flowchart* untuk rancangan alur sistem dari aplikasi yang akan dibuat dengan menggunakan arsitektur *microservices*. Pada Gambar 4.2 dapat dilihat pada saat pengguna mulai menggunakan aplikasi, maka diperlukan konfigurasi Firebase *Authentication* terlebih dahulu pada bagian Android, kemudian menambahkan dependensi Firebase *Authentication*¸ melakukan inisialisasi Firebase, dan mendapatkan *instance* Firebase *Authentication*.

Jika pengguna sudah memiliki akun, maka pengguna hanya perlu masuk dengan akun yang telah didaftarkan kemudian pengguna akan mendapatkan token autentikasi. Namun jika pengguna belum memiliki akun, maka pengguna perlu melakukan registrasi akun dengan mendaftarkan akun pengguna, kemudian pengguna akan mendapatkan token autentikasi untuk dimasukkan dan sistem akan memproses untuk verifikasi token. Jika proses autentikasi tidak berhasil maka pengguna akan menerima status autentikasi gagal dan program akan berakhir, namun jika berhasil maka pengguna akan masuk ke menu utama aplikasi.

Pengguna dapat menentukan apa yang ingin dilakukan terhadap fungsi-fungsi yang ada dalam aplikasi, jika pengguna ingin *sign out* atau keluar dari aplikasi maka pengguna perlu melakukan proses *sign out* dan sistem akan menghapus token autentikasi pengguna maka program selesai.

Jika pengguna tidak ingin keluar dari aplikasi maka sistem akan tetap menjalankan aplikasi. Pengguna kemudian dapat memilih beberapa menu seperti profil, klasifikasi, analisis, dan riwayat analisis. Setiap proses yang terjadi di

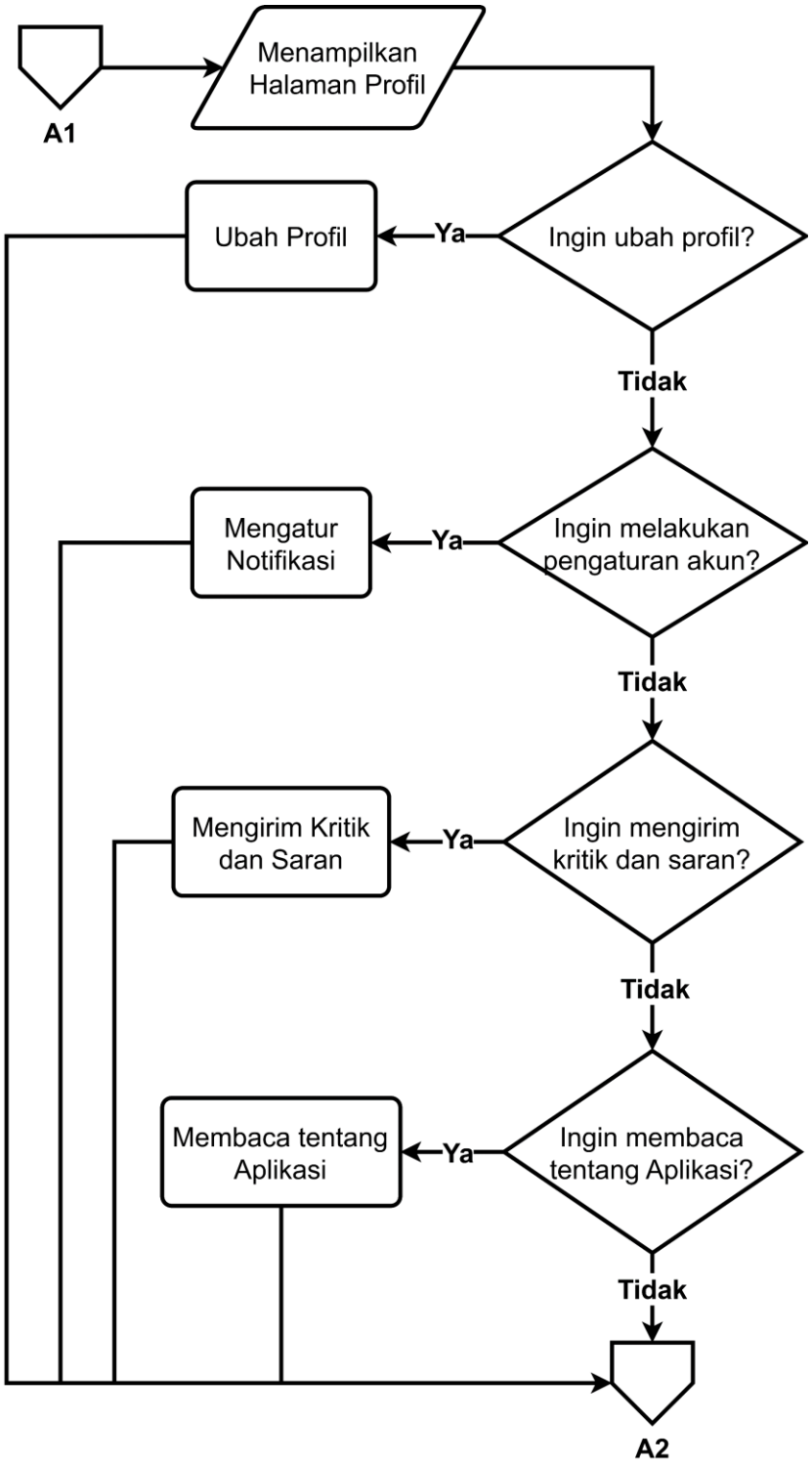
menu akan berakhir di menu utama aplikasi, maka pengguna dapat menentukan kembali apakah ingin keluar dari aplikasi atau tidak dan program selesai.



Gambar 4.2 *Flowchart* Utama Aplikasi

Pada Gambar 4.2 dapat dilihat simbol *off-page reference* A1, A2, B1, B2 dan simbol *on-page reference* C1. Simbol *off-page reference* A1 merujuk pada

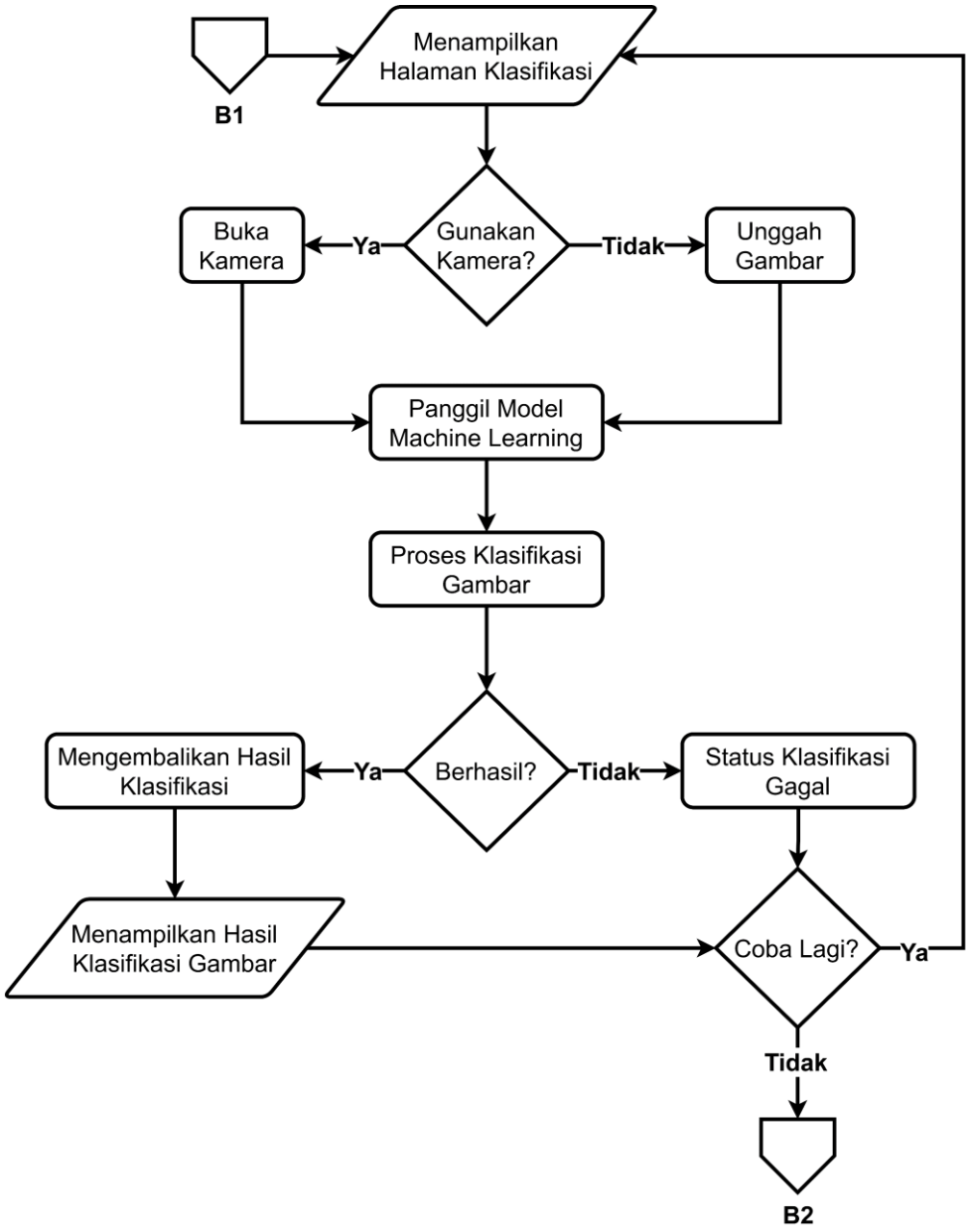
Gambar 4.3 di bawah ini. Jika pada Gambar 4.2 pengguna memilih untuk navigasi ke Profil, maka program akan menampilkan proses pada Gambar 4.3.



Gambar 4.3 *Flowchart* A1 Menu Profil

Pada Gambar 4.3 menunjukkan *flowchart* untuk menampilkan menu profil pengguna. Pada alur diatas, pengguna dapat memilih beberapa proses seperti mengubah profil, mengatur notifikasi, mengirim kritik dan saran, atau membaca tentang aplikasi. Seluruh proses yang terjadi akan membawa pengguna ke simbol *off-page reference* A2, yaitu navigasi ke menu utama aplikasi yang dapat dilihat pada Gambar 4.2.

Sedangkan simbol *off-page reference* B1 merujuk pada Gambar 4.4 di bawah ini. Jika pada Gambar 4.2 pengguna memilih untuk melakukan klasifikasi gambar, maka program akan menampilkan proses pada Gambar 4.4.



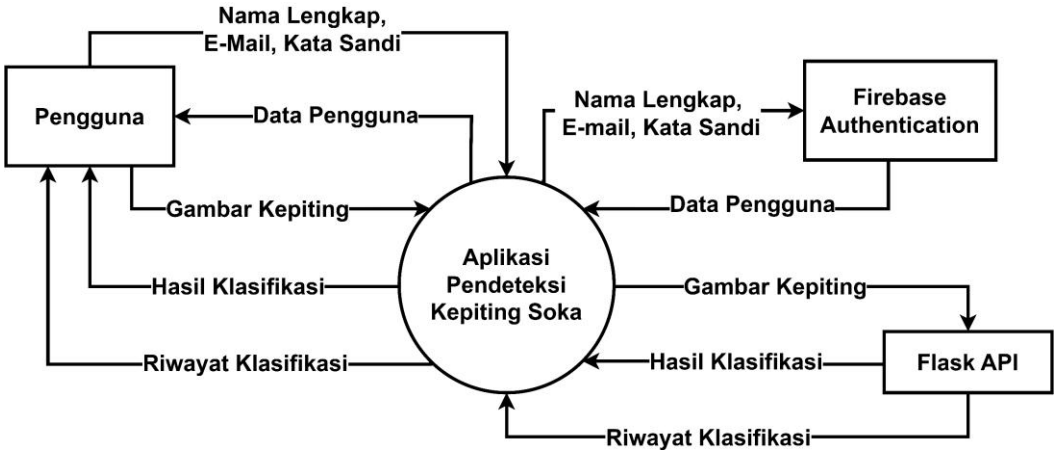
Gambar 4.4 *Flowchart* B1 Klasifikasi Gambar

Gambar 4.4 merupakan *flowchart* untuk proses menampilkan menu klasifikasi gambar. Pada menu ini pengguna dapat menentukan untuk melakukan klasifikasi gambar dengan menggunakan kamera secara langsung atau dengan mengunggah gambar yang akan diklasifikasikan dalam hal ini adalah gambar kepiting.

Jika pengguna memilih untuk menggunakan kamera, kemudian program akan melakukan proses untuk mengakses kamera, kemudian program akan memanggil model *machine learning* dan langsung memproses klasifikasi gambar. Begitu juga jika pengguna memilih untuk mengunggah gambar, maka setelah program memproses gambar, kemudian program akan memanggil model *machine learning* dan langsung memproses klasifikasi gambar. Jika proses berhasil, maka pengguna dapat melihat hasil klasifikasi gambar. Namun jika proses tidak berhasil, maka pengguna akan menerima status klasifikasi gagal. Setelah proses tersebut pengguna dapat menentukan untuk mencoba lagi proses sebelumnya atau tidak. Jika pengguna memilih ya, maka proses akan diulang ke menampilkan menu klasifikasi gambar. Namun jika tidak, maka proses akan diarahkan ke simbol *off-page reference* B2, yaitu navigasi ke menu utama aplikasi yang dapat dilihat pada Gambar 4.2.

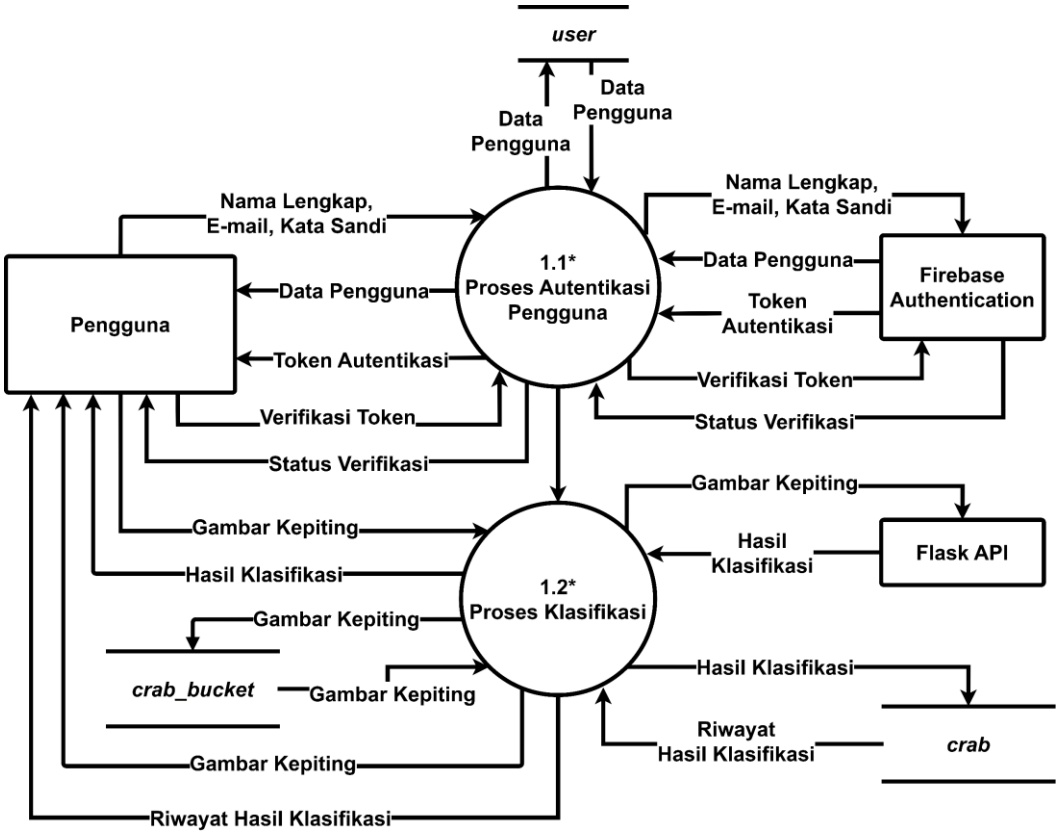
4.2.1.2. *Data Flow Diagram*

Pada Gambar 4.5 dan Gambar 4.6 menunjukkan pemodelan *data flow diagram* dari aplikasi *back-end* menggunakan arsitektur *microservice* yang akan dibuat.



Gambar 4.5 Diagram DFD Level 0

Data flow diagram pada Gambar 4.5 dimulai saat pengguna memberikan *input* berupa nama lengkap, *e-mail*, dan kata sandi dan melalui proses yang ada dalam aplikasi pendeteksi kepiting soka *input* tersebut akan dikirim ke Firebase *Authentication* untuk dapat dikembalikan ke pengguna sebagai *output* data pengguna. Setelah itu, pengguna bisa memberikan *input* berupa gambar kepiting dan melalui proses yang ada dalam aplikasi juga gambar tersebut akan dikirim ke Flask API dan di proses menggunakan model *machine learning* yang telah diimplementasikan di dalam Flask API untuk memberikan *output* berupa hasil klasifikasi dan riwayat hasil klasifikasi kepada pengguna.

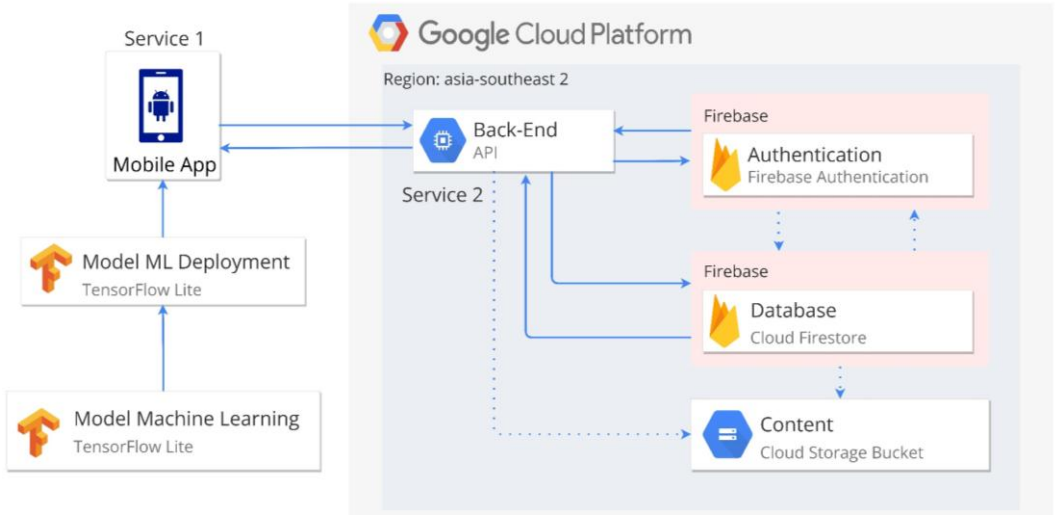


Gambar 4.6 Diagram DFD Level 1

Data flow diagram pada Gambar 4.6 menunjukkan proses alur data yang lebih rinci dari setiap proses yang terjadi di dalam aplikasi pendeteksi kepiting soka yang akan dibuat.

4.2.2. Desain Arsitektur *Microservices*

Pada Gambar 4.7 menunjukkan desain arsitektur *microservices* yang akan digunakan sebagai dasar saat melakukan tahap implementasi nanti. Model *machine learning* yang di *deploy* secara langsung ke dalam aplikasi untuk memudahkan pengguna dalam menggunakan aplikasi secara *offline*. Aplikasi Pendeteksi Kepiting Soka juga terhubung ke *back-end* dengan menggunakan arsitektur *microservices* dari *Google Cloud Platform* untuk membuat aplikasi dapat berjalan secara *online* pada *region asia-southeast*-2. Kemudian dihubungkan ke *Firebase* untuk melakukan autentikasi pengguna, menyimpan data dalam *Cloud Firestore* dan menyimpan gambar menggunakan *Cloud Storage Bucket* dalam *Google Cloud Platform*.



Gambar 4.7 Arsitektur *Microservices*

4.3. *Development*

Pada bagian ini, akan diuraikan proses implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka. Proses implementasi ini melibatkan pemisahan fungsionalitas aplikasi menjadi layanan-layanan mandiri yang dapat beroperasi secara independen, meningkatkan modularitas, dan memfasilitasi pengembangan, penyebaran, serta pemeliharaan yang lebih efektif.

4.3.1. Lingkungan Implementasi

Pada bagian ini, akan diuraikan secara terperinci mengenai lingkungan implementasi perangkat keras dan perangkat lunak yang digunakan selama melakukan tahap pengembangan arsitektur *microservice* pada Aplikasi Pendeteksi

Kepiting Soka. Rincian ini mencakup spesifikasi perangkat keras yang diperlukan untuk mendukung proses pengembangan serta perangkat lunak yang digunakan, seperti sistem operasi, perangkat lunak pengembangan (IDE), dan alat bantu pengujian, juga akan dijabarkan untuk memberikan gambaran menyeluruh tentang lingkungan kerja yang digunakan selama tahap *development*.

4.3.1.1. Perangkat Keras

Pada Tabel 4.1 menunjukkan spesifikasi perangkat keras yang digunakan untuk mengimplementasikan arsitektur *microservice*.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Nama Perangkat Keras	Spesifikasi
Lenovo IdeaPad S145-14IWL	*Processor* : Intel(R) Celeron(R) CPU 4205U @ 1.80GHz 1.80 GHz *Installed* RAM: 4.00 GB (3.88 GB usable) *System type* : 64-bit *operating system*, x64-*based processor*

4.3.1.2. Perangkat Lunak

Pada Tabel 4.2 menunjukkan daftar perangkat lunak yang digunakan untuk membuat arsitektur *microservices*.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

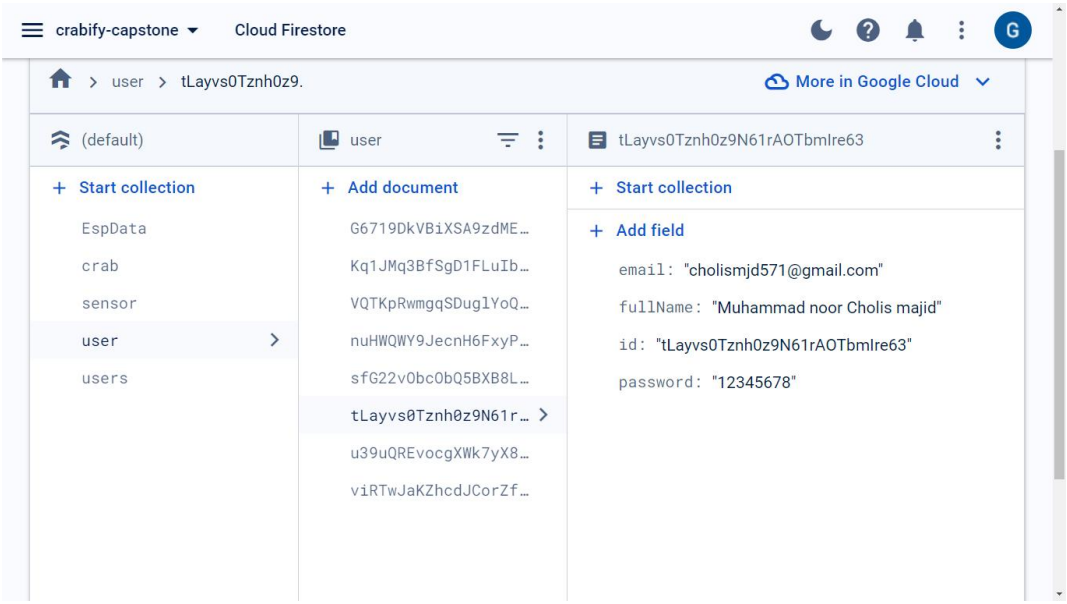
Nama Perangkat Lunak	Spesifikasi/Kegunaan
Device Operating System	*Edition* : Windows 11 *Education* *Version* : 22H2 *Installed on* : 10/9/2022 OS *build* : 22621.3007 *Serial Number* : PF1Q6S81
Google Cloud Platform	Menyimpan data gambar pada *Cloud Storage Bucket*.
Firebase	Menyimpan data *realtime*, melakukan autentikasi pengguna, dan melakukan analisis data untuk membantu pengembang membangun dan mengelola aplikasi dengan lebih mudah.
Draw.io	Membuat diagram alur sistem, dan diagram alur data.

4.3.2. Implementasi Arsitektur *Microservices*

Dalam implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka, hasilnya mencakup beberapa aspek yang menunjukkan keberhasilan dan keuntungan dari pendekatan ini. Berikut adalah rincian hasil implementasi arsitektur *microservice* berupa *cloud firestore*, *cloud storage bucket*, dan TensorFlow yang dibuat menggunakan Flask API.

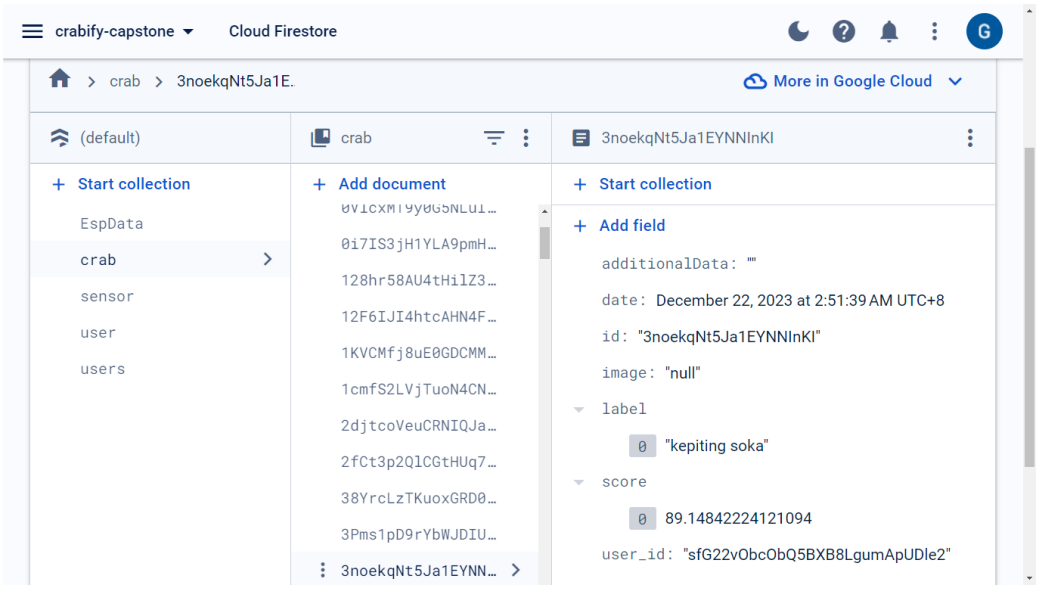
4.3.2.1. Implementasi *Cloud Firestore*

Pada Gambar 4.8 dan Gambar 4.9 menunjukkan hasil implementasi *database cloud firestore* yang digunakan dalam pengembangan aplikasi ini. Keberadaan layanan *microservice Cloud Firestore* berperan penting dengan menyediakan infrastruktur yang handal untuk menyimpan data pengguna. Dengan sinergi antara Firebase *Authentication* dan *Cloud Firestore*, aplikasi ini dapat memberikan pengalaman pengguna yang lebih baik melalui sistem autentikasi yang aman dan manajemen data yang efisien.



Gambar 4.8 Implementasi *Database* Tabel *User*

Pada Gambar 4.8 menunjukkan tabel *user* pada *database crabify-capstone*. Dalam tabel *user* terdapat beberapa data pengguna yang sudah terdaftar dalam aplikasi pendeteksi kepiting soka. Dalam setiap dokumen memiliki atribut *e-mail*, nama lengkap (*fullName*), *Universally Unique Identifier* (UUID), dan kata sandi (*password*).



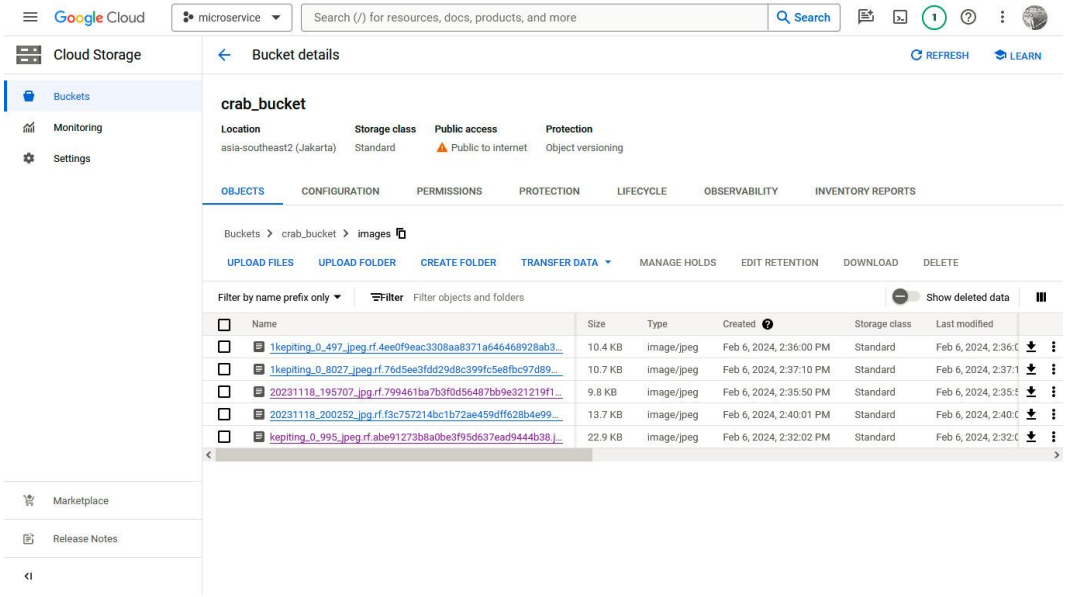
Gambar 4.9 Implementasi *Database* Tabel *Crab*

Pada Gambar 4.9 menunjukkan tabel *crab* pada *database crabify-capstone*. Dalam tabel *crab* terdapat beberapa data klasifikasi gambar yang sudah diuji coba melalui aplikasi pendeteksi kepiting soka kemudian menyimpan hasilnya ke dalam tabel *crab* diatas. Dalam setiap dokumen memiliki atribut tanggal (*date*), *Universally Unique Identifier* (UUID), gambar (*image*), label, tingkat akurasi (*score*), dan *user_id*.

4.3.2.2. Implementasi *Cloud Storage Bucket*

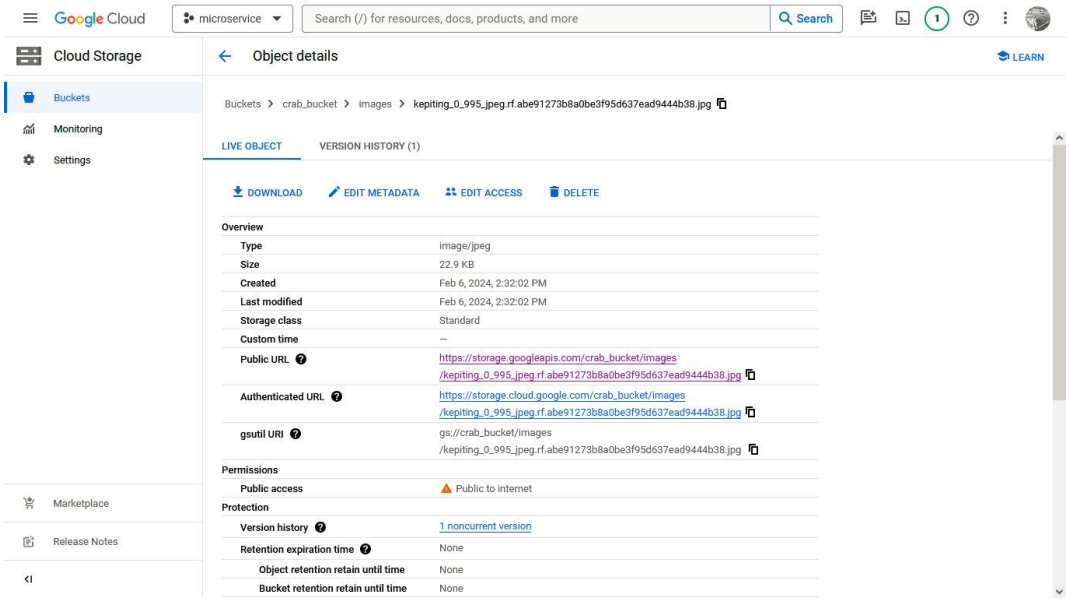
Pada Gambar 4.10 menunjukkan hasil implementasi *cloud storage bucket* yang digunakan untuk menyimpan data gambar. Penerapan *microservice cloud storage bucket* memberikan pengembang kemudahan dalam menyediakan penyimpanan aman dan *scalable* untuk berbagai jenis *file* dan data. *Cloud Storage Bucket* memungkinkan pengembang untuk menyimpan objek-objek biner seperti gambar, *video*, atau dokumen dengan efisien di lingkungan *Google Cloud Storage*. *Bucket*

menyediakan infrastruktur yang handal dan terkelola dengan baik, memastikan ketersediaan data yang tinggi dan skalabilitas untuk menangani kebutuhan penyimpanan yang beragam.



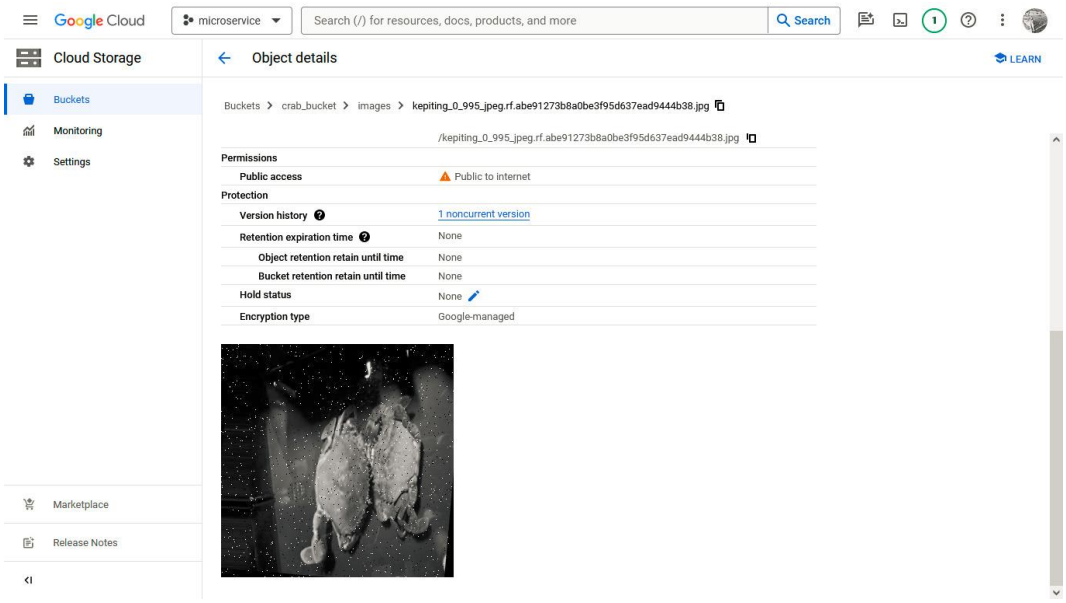
Gambar 4.10 Implementasi *Cloud Storage Bucket*

Integrasi *Cloud Storage Bucket* dengan aplikasi memungkinkan pengguna untuk menyimpan, berbagi, dan mengelola *file-file* mereka dengan mudah. Dengan penyimpanan *file* yang efisien, aplikasi dapat memberikan pengalaman pengguna yang lebih baik dan responsif. Pengembang dapat menggunakan antarmuka Firebase *Storage* untuk berinteraksi dengan *Cloud Storage Bucket* dan mengelola objek-objek dalam penyimpanan dengan lebih fleksibel.



Gambar 4.11 Rincian Objek Tersimpan

Gambar 4.11 dan Gambar 4.12 memberikan gambaran rinci tentang objek yang tersimpan dalam *Cloud Storage Bucket* ketika pengguna aktif melakukan proses klasifikasi gambar melalui aplikasi pendeteksi kepiting soka. Pengguna, meskipun tidak memiliki akses langsung ke *Cloud Storage Bucket*, dapat mempercayakan penyimpanan gambar-gambar tersebut di sini. Keputusan untuk menyimpan hasil klasifikasi secara terpusat dalam wadah *Cloud Storage Bucket* didasarkan pada pertimbangan strategis untuk memudahkan pengelolaan dan administrasi media yang telah dikumpulkan oleh aplikasi.

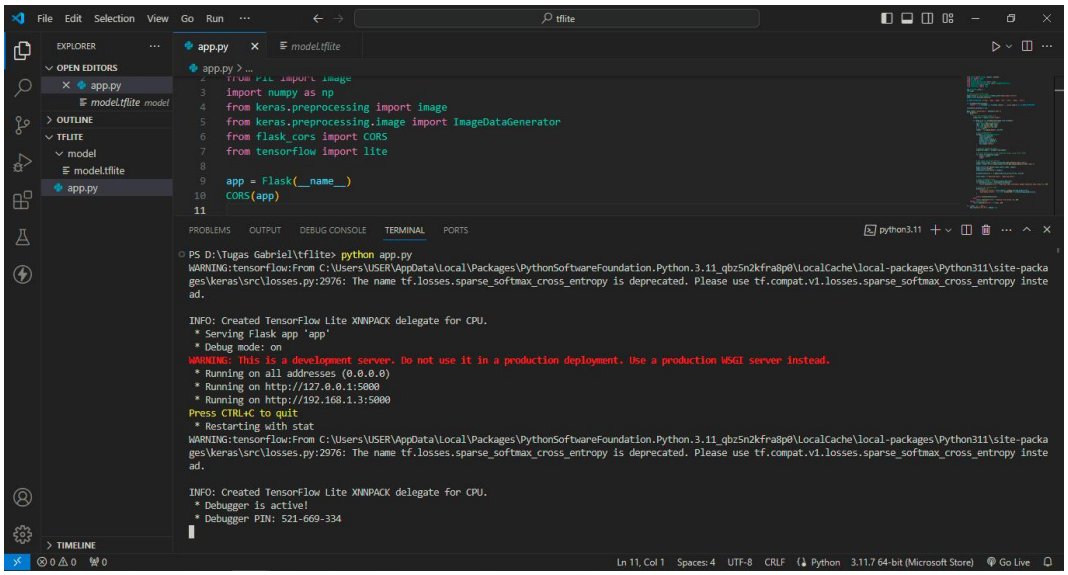


Gambar 4.12 Lanjutan - Rincian Objek Tersimpan

Pentingnya pengelolaan yang efisien dari segi penyimpanan dan ketersediaan media menjadi faktor utama dalam desain arsitektur penyimpanan. Dengan demikian, pengguna dapat dengan nyaman memanfaatkan hasil klasifikasi gambar untuk keperluan analisis atau referensi di masa mendatang. Meskipun secara langsung tidak dapat mengakses kontennya dalam *Cloud Storage Bucket*, pengguna dapat mengandalkan aplikasi untuk memberikan aksesibilitas dan fungsionalitas yang sesuai dengan kebutuhan pengelolaan media yang dikelola di belakang layar. Dengan pendekatan ini, aplikasi pendeteksi kepiting soka menciptakan lingkungan yang efisien, handal, dan mudah dikelola, meningkatkan pengalaman pengguna secara menyeluruh.

4.3.2.3. Implementasi TensorFlow *Lite* pada Flask API

Berikut ini merupakan hasil implementasi TensorFlow *Lite* Flask API yang digunakan sebagai *route* untuk memanggil model *machine learning*, sehingga aplikasi dapat terhubung dengan model *machine learning* melalui API ini.



Gambar 4.13 *Flask App Running*

Gambar 4.13 menunjukkan bahwa API Flask *app* dapat berjalan dengan baik pada *localhost* 127.0.0.1:5000 dan Alamat IP 192.168.1.3:5000. Pada *localhost* 127.0.0.1:5000, Flask *app* diikat pada alamat IP *loopback* 127.0.0.1, yang secara eksklusif merujuk pada mesin lokal tempat aplikasi dijalankan. *Port* yang digunakan adalah 5000, yang merupakan *port default* Flask jika tidak ditentukan secara eksplisit. Keuntungannya adalah berguna saat pengembangan lokal atau pengujian aplikasi, namun, keterbatasannya adalah bahwa aplikasi hanya dapat diakses oleh mesin tempat aplikasi berjalan.

Pada alamat IP 192.168.1.3:5000, Flask *app* diikat pada alamat IP 192.168.1.3 yang terkait dengan antarmuka jaringan pada mesin tersebut. Ini memungkinkan akses ke Flask *app* dari perangkat lain di jaringan lokal, berguna untuk pengujian dalam lingkup LAN. Keterbatasannya adalah aplikasi masih terbatas pada jaringan tersebut dan dapat diakses oleh perangkat lain di jaringan, namun, tidak dari internet secara umum.

4.3.3. Implementasi Modul Program

Tabel 4.3 merupakan daftar dari beberapa modul program yang telah diimplementasikan dalam aplikasi yang dibuat. Pada Tabel 4.3 ini memberikan gambaran mengenai modul-modul program yang terdapat dalam aplikasi. Setiap modul memiliki fungsi dan tanggung jawab tertentu, menciptakan kerangka kerja

yang terstruktur dan terorganisir untuk mendukung fungsionalitas keseluruhan aplikasi.

Tabel 4.3 Implementasi Modul Program

Nama Modul	Kode Program
Modul Flask dan Modul Pendukung	from flask import Flask, request, jsonify from flask_cors import CORS
Modul Pengolahan Gambar	from PIL import Image import numpy as np from keras.preprocessing import image from keras.preprocessing.image import ImageDataGenerator
Modul TensorFlow *Lite*	from tensorflow import lite
Inisialisasi Aplikasi Flask dan Konfigurasi CORS	app = Flask(__name__) CORS(app)
Inisialisasi dan Alokasi Model *TFLite*	model_tflite = lite.Interpreter(model_path="model\model.tflite") model_tflite.allocate_tensors()
Variabel dan fungsi pendukung	ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'tiff', 'webp', 'jfif'} def allowed_file(filename): return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS confidence_threshold = 0.8 posted_data = []

Rute untuk Prediksi (*POST Request*)	@app.route('/klasifikasi', methods=['POST'])
	def predict():
	  global posted_data # Gunakan global untuk menyimpan data di-POST
	  try:
	  # Get the uploaded image file
	  image_file = request.files['image']

Nama Modul	Kode Program

	if image_file and allowed_file(image_file.filename):
	# Prepare image for prediction
	img = Image.open(image_file)
	img = img.resize((300, 300))
	x = image.img_to_array(img)
	x = x / 255.0
	images = np.expand_dims(x, axis=0)
	
	# Data augmentation
	datagen = ImageDataGenerator(
	rotation_range=40,
	shear_range=0.2,
	width_shift_range=0.2,
	height_shift_range=0.2,
	horizontal_flip=True,
	fill_mode='nearest'
	)
	
	# Generate augmented images
	augmented_images = datagen.flow(images)
	
	# Perform predictions on the augmented images using TFLite model
	for batch in augmented_images:
	inputs = batch
	break
	
	# Get input and output tensors
	input_tensor_index = model_tflite.get_input_details()[0]['index']
	output = model_tflite.tensor(model_tflite.get_output_details()[0]['index'])

Nama Modul	Kode Program

	model_tflite.set_tensor(input_tensor_index, inputs)
	model_tflite.invoke()
	prediction_array_tflite = output()
	
	average_prediction = np.mean(prediction_array_tflite, axis=0)
	
	class_names = ['Kepiting Biasa', 'Kepiting Soka']
	
	# Check confidence level
	confidence_tflite = np.max(average_prediction)
	if confidence_tflite < confidence_threshold:
	return jsonify({"error": "Kepiting tidak terdeteksi dengan keyakinan yang cukup."}), 400
	
	# Format the response JSON
	predictions = {
	"prediction_tflite": class_names[np.argmax(average_prediction)],
	"confidence_tflite": '{:2.0f}%'.format(100 * np.max(average_prediction)),
	}
	
	# Simpan data yang di-POST ke variabel global
	posted_data.append(predictions)
	
	return jsonify(predictions)
	else:
	return jsonify({"error": "Invalid file format."}), 400
	except Exception as e:
	return jsonify({"error": str(e)}), 500

Rute untuk *GET Data* yang di POST	@app.route('/get_data', methods=['GET']) def get_posted_data():

Nama Modul	Kode Program

	global posted_data
	if posted_data:
	return jsonify({"posted_data": posted_data})
	else:
	return jsonify({"message": "No data has been posted yet."})

Rute untuk *methods GET All Data*	@app.route('/crabify', methods=['GET']) def hello(): return jsonify({"message": "Hello, this is a GET request!"})

4.4. *Testing*

Pada tahap ini, aplikasi yang telah berhasil dibangun akan menjalani serangkaian uji coba yang dirancang untuk menilai dan menentukan kualitas dari implementasi arsitektur *microservice* pada Aplikasi Pendeteksi Kepiting Soka. Implementasi arsitektur *microservice* pada aplikasi ini merupakan suatu pendekatan yang memecah monolitik menjadi layanan-layanan terpisah, masing-masing bertanggung jawab atas fungsionalitas tertentu.

4.4.1. Tujuan Pengujian

Pengujian ini tidak hanya bertujuan untuk memastikan bahwa aplikasi dapat berjalan dengan baik, tetapi juga memiliki tujuan lebih luas yaitu untuk menganalisis kemungkinan kegagalan terhadap fungsionalitasnya. Dengan melakukan serangkaian pengujian yang komprehensif, pengembang dapat mengidentifikasi potensi masalah atau cacat dalam aplikasi. Penting untuk menangkap dan memperbaiki kegagalan ini sebelum aplikasi didistribusikan kepada pengguna akhir.

Pengujian fungsionalitas membantu dalam mengamati perilaku aplikasi dalam berbagai skenario penggunaan dan memverifikasi apakah fungsi-fungsi inti berjalan sesuai dengan yang diharapkan. Hal ini melibatkan uji coba setiap fitur dan komponen aplikasi, termasuk integrasi antar komponen serta interaksi dengan pengguna.

Dengan melakukan pengujian ini, tim pengembang dapat mengidentifikasi dan memperbaiki masalah sejak dini, mengurangi risiko kegagalan setelah aplikasi diimplementasikan di lingkungan produksi. Hal ini akan membantu meningkatkan kualitas dan keandalan aplikasi secara keseluruhan, memberikan pengalaman yang lebih baik bagi pengguna, dan meningkatkan kepercayaan terhadap aplikasi yang didistribusikan.

4.4.2. Kriteria Pengujian

Berikut ini merupakan beberapa kriteria pengujian yang diharapkan dapat tercapai dengan melakukan tahap pengujian ini.

1. Implementasi arsitektur *microservice* melalui Flask API dapat memberikan respons yang baik dan cepat saat pengguna mengirimkan permintaan.
2. Implementasi arsitektur *microservice* melalui Flask API dapat memproses model *machine learning* sehingga pengguna dapat menggunakan fitur untuk klasifikasi gambar.
3. Implementasi arsitektur *microservice* melalui Flask API dapat memberikan respons jika terjadi kegagalan pada saat menggunakan fitur klasifikasi.
4. Implementasi arsitektur *microservice* melalui Flask API dapat menyimpan hasil klasifikasi pada *Cloud Storage Bucket*.

4.4.3. Kasus Pengujian

Pada Tabel 4.4 diberikan beberapa kasus pengujian yang akan dilakukan untuk menguji arsitektur *microservice* yang dibuat dengan menggunakan Flask API.

Tabel 4.4 Kasus Pengujian

No	Kasus Pengujian
1.	Apakah arsitektur *microservice* yang di implementasikan melalui Flask API dapat memberikan respons yang sesuai dan cepat saat menerima permintaan *client*?
2.	Apakah arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memproses model *machine learning* dan memberikan klasifikasi yang sesuai?
3.	Apakah arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika format gambar yang dimasukkan

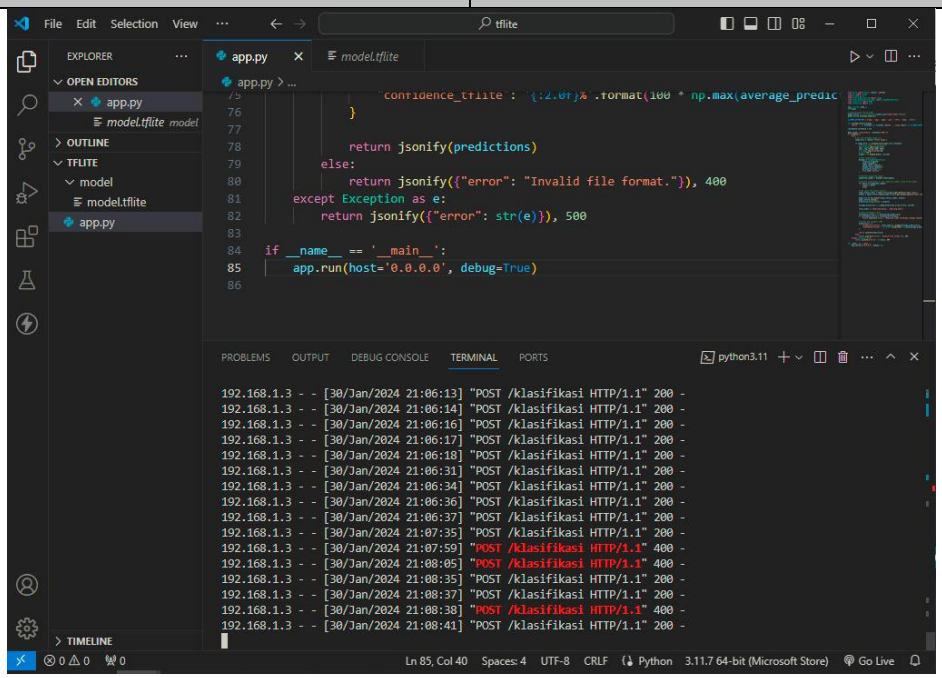
	tidak sesuai?
4.	Apakah arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika gambar yang dimasukkan tidak termasuk dalam kelas klasifikasi dalam model *machine learning*?
5.	Apakah arsitektur *Cloud Firestore* yang di implementasikan melalui Flask API dapat menyimpan hasil klasifikasi?
6.	Apakah arsitektur *Cloud Storage Bucket* yang di implementasikan melalui Flask API dapat menyimpan riwayat klasifikasi gambar?
7.	Apakah arsitektur *microservice* yang di implementasikan melalui Flask API dapat menjalankan metode *GET* untuk mendapatkan riwayat klasifikasi?

4.4.4. Pelaksanaan Pengujian

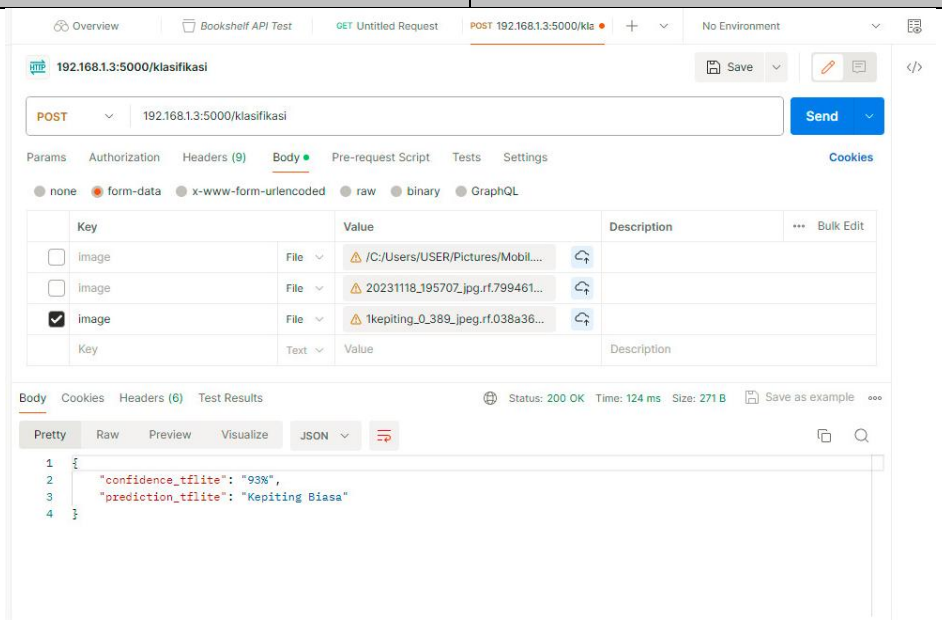
Tabel 4.5 merupakan dokumentasi hasil pengujian dari kasus pengujian yang dilakukan. merupakan hasil pengujian yang dilakukan terhadap kasus pengujian yang telah disebutkan sebelumnya.

Tabel 4.5 Pelaksanaan Pengujian

No.	Kasus Pengujian	Hasil Pengujian
1.	Apakah arsitektur *microservice* yang di implementasikan melalui Flask API dapat memberikan respons yang sesuai dan cepat saat menerima permintaan *client*?	Arsitektur *microservice* yang di implementasikan melalui Flask API dapat memberikan respons yang sesuai dan cepat saat menerima permintaan *client.*
	Pada gambar 4.14 menunjukkan bahwa pengujian API dapat berjalan dengan baik. Dapat dilihat ketika *client* melakukan permintaan ke server. API dapat memberikan respons yang sesuai dan cepat dalam beberapa detik untuk memproses permintaan pengguna.	

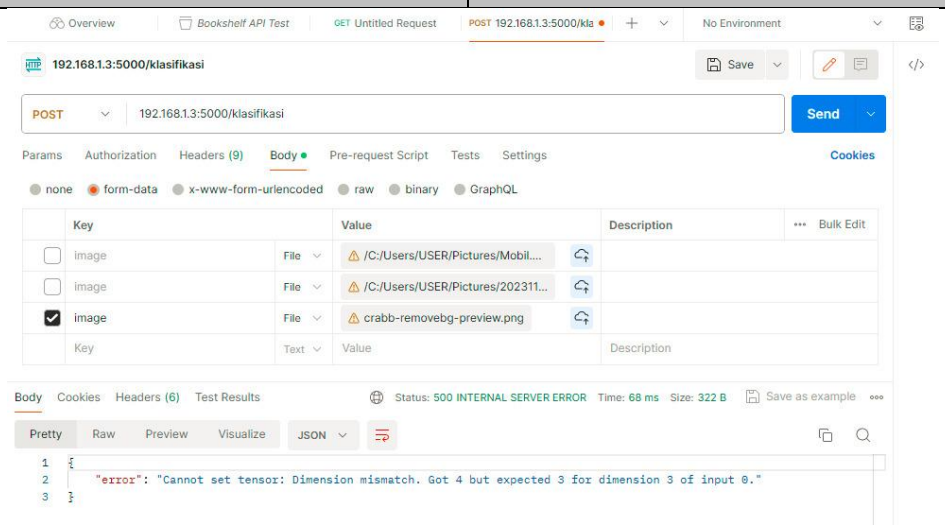
<table>
<tr><th>No.</th><th colspan="2">Kasus Pengujian</th><th>Hasil Pengujian</th></tr>
<tr><td></td><td colspan="3"> 

Gambar 4.14 Hasil Pengujian Kecepatan Respons API</td></tr>
<tr><td>2.</td><td colspan="2">Apakah arsitektur microservice TensorFlow yang di implementasikan melalui Flask API dapat memproses model machine learning dan memberikan klasifikasi yang sesuai?</td><td>Arsitektur microservice TensorFlow yang di implementasikan melalui Flask API dapat memproses model machine learning dan memberikan klasifikasi yang sesuai.</td></tr>
<tr><td></td><td colspan="3">Gambar 4.15 memberikan representasi yang baik terkait keberhasilan pengujian API menggunakan Postman. Secara khusus, pengujian POST pada alamat IP 192.168.1.3:5000 menunjukkan bahwa program memberikan respons yang sangat memuaskan. Hasil klasifikasi gambar diterima dalam format JSON, memberikan informasi yang terstruktur dan mudah diinterpretasi. Pada hasil klasifikasi tersebut, terlihat tingkat kepercayaan (confidence level) mencapai 93%, menandakan bahwa model klasifikasi bekerja dengan sangat baik dalam mengidentifikasi dan mengklasifikasikan gambar yang diuji.</td></tr>
</table>

<table>
<tr><th>No.</th><th>Kasus Pengujian</th><th>Hasil Pengujian</th></tr>
<tr><td></td><td colspan="2"> 

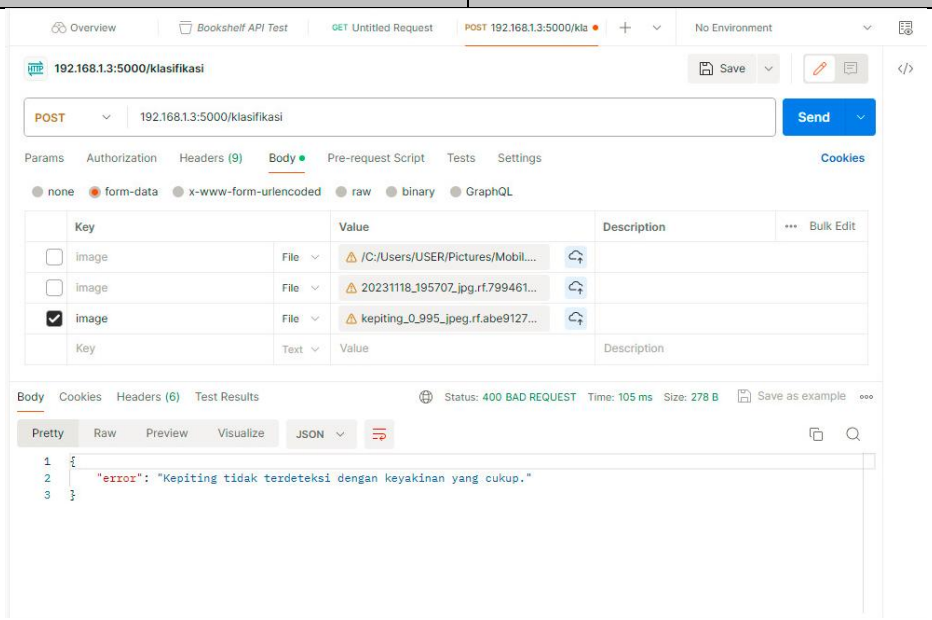
Gambar 4.15 Hasil Pengujian API

Keberhasilan pengujian ini juga menyoroti kemampuan API untuk memberikan respons yang cepat dan informatif. Respons yang efisien sangat penting, terutama dalam konteks aplikasi klasifikasi gambar, di mana penggunaan model *machine learning* dan pemrosesan gambar dapat memerlukan waktu yang signifikan. Dengan demikian, gambaran keseluruhan dari pengujian ini memberikan keyakinan bahwa API siap untuk diterapkan dalam produksi, memberikan hasil klasifikasi gambar yang akurat dan responsif bagi pengguna.</td></tr>
<tr><td>3.</td><td>Apakah arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika format gambar yang dimasukkan tidak sesuai?</td><td>Arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika format gambar yang dimasukkan tidak sesuai.</td></tr>
<tr><td></td><td colspan="2">Gambar 4.16 dalam pengujian API menggunakan Postman membuktikan kelancaran jalannya uji coba terkait kegagalan ketika model mendeteksi gambar yang dimasukkan tidak sesuai dengan format yang telah ditetapkan. Dalam skenario ini, melalui uji *POST* pada alamat IP 192.168.1.3:5000, respons yang dihasilkan oleh program menunjukkan hasil klasifikasi dengan status *error*. Situasi ini mengindikasikan bahwa model klasifikasi tidak mampu mengenali atau memproses gambar yang diinput sesuai dengan format yang diharapkan.</td></tr>
</table>

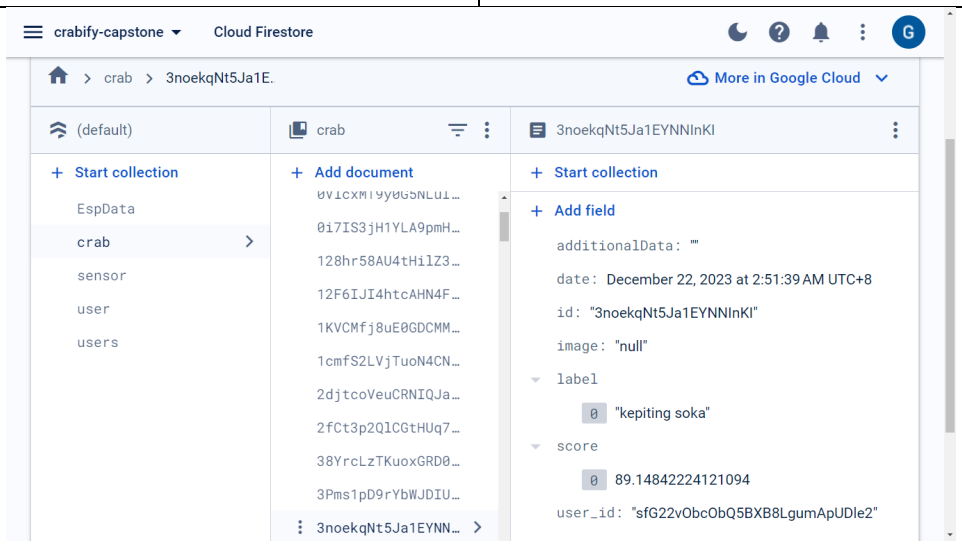
<table>
<tr><th>No.</th><th>Kasus Pengujian</th><th>Hasil Pengujian</th></tr>
<tr><td></td><td colspan="2"> 

Gambar 4.16 Hasil Pengujian *Output Error* 400

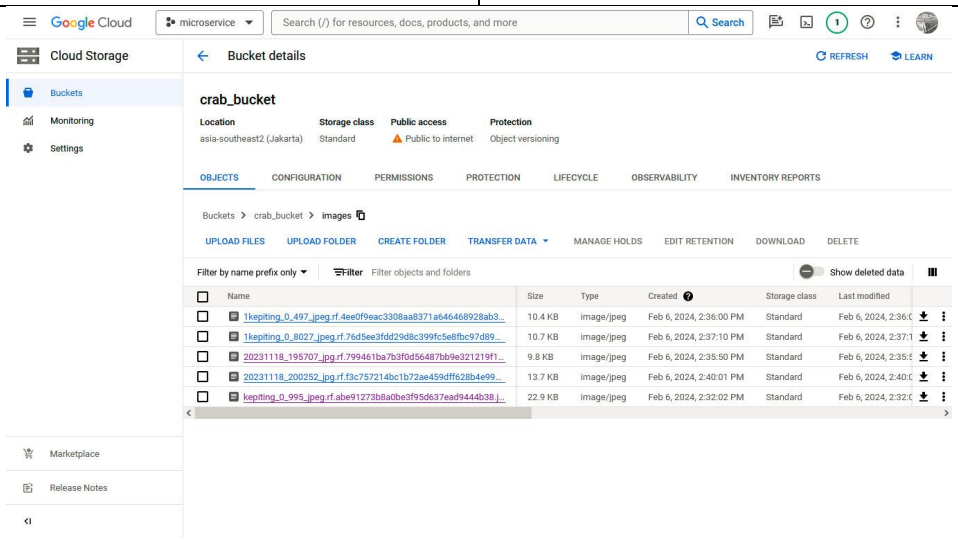
Penting untuk dicatat bahwa format gambar yang diharapkan oleh model memainkan peran kritis dalam keberhasilan pengklasifikasian. Kegagalan yang tercatat dalam respons JSON menggambarkan bahwa model menghadapi kesulitan dalam menafsirkan struktur atau karakteristik tertentu yang diperlukan sesuai dengan standar format yang ditentukan.</td></tr>
<tr><td rowspan="2">4.</td><td>Apakah arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika gambar yang dimasukkan tidak termasuk dalam kelas klasifikasi dalam model *machine learning*?</td><td>Arsitektur *microservice* TensorFlow yang di implementasikan melalui Flask API dapat memberikan respons gagal jika gambar yang dimasukkan tidak termasuk dalam kelas klasifikasi dalam model *machine learning.*</td></tr>
<tr><td colspan="2">Gambar 4.17 menunjukkan pengujian menggunakan Postman memberikan gambaran yang sangat informatif terkait keberhasilan pengujian kegagalan ketika model tidak dapat mengenali gambar yang dimasukkan sebagai bagian dari salah satu kelas klasifikasinya. Melalui uji *POST* yang dilakukan pada alamat IP 192.168.1.3:5000, program memberikan respons dalam format JSON yang mengindikasikan hasil klasifikasi sebagai *error*. Kondisi ini menunjukkan bahwa model klasifikasi tidak dapat memberikan prediksi atau mengidentifikasi kategori klasifikasi gambar yang diberikan.</td></tr>
</table>

<table>
<tr><th>No.</th><th>Kasus Pengujian</th><th>Hasil Pengujian</th></tr>
<tr><td></td><td colspan="2"> 

Gambar 4.17 Hasil Pengujian *Output Error* 500

Perlu diperhatikan bahwa kemampuan model untuk mengklasifikasikan gambar sangat dipengaruhi oleh sejumlah faktor, dan kegagalan yang terlihat pada hasil pengujian ini dapat disebabkan oleh berbagai alasan. Beberapa faktor yang mungkin mempengaruhi kegagalan ini melibatkan karakteristik gambar yang diuji, seperti kualitas gambar yang rendah, ketidakjelasan objek, atau adanya gangguan visual yang dapat mengaburkan identifikasi kelas oleh model.</td></tr>
<tr><td>5.</td><td>Apakah arsitektur *Cloud Firestore* yang di implementasikan melalui Flask API dapat menyimpan hasil klasifikasi?</td><td>Arsitektur *Cloud Firestore* yang di implementasikan melalui Flask API dapat menyimpan hasil klasifikasi.</td></tr>
<tr><td></td><td colspan="2"> 
</td></tr>
</table>

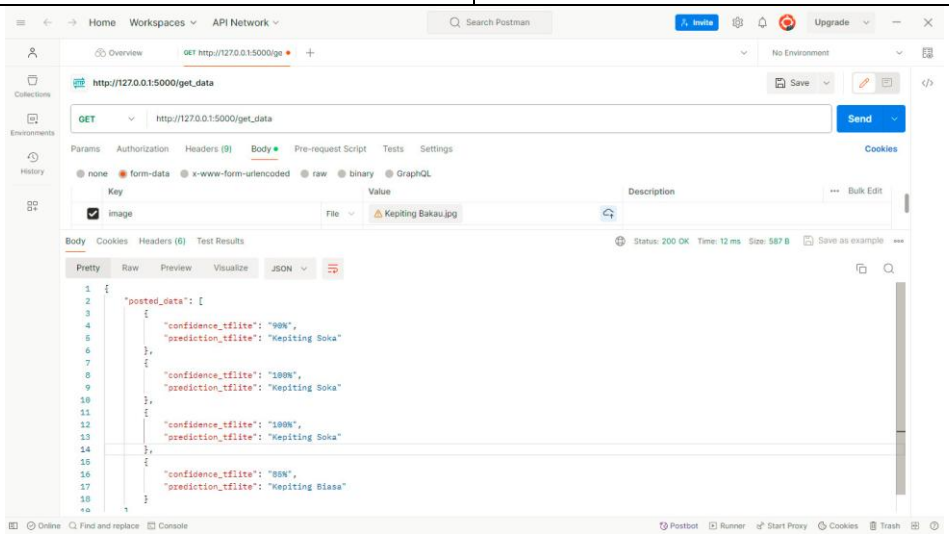
<table>
<tr><th>No.</th><th>Kasus Pengujian</th><th>Hasil Pengujian</th></tr>
<tr><td></td><td colspan="2">Gambar 4. 18 Hasil Pengujian Simpan Data di Firestore

Gambar 4.18 menunjukkan hasil pengujian pada arsitektur Cloud Firestore yang di implementasikan melalui Flask API dapat menyimpan hasil klasifikasi dalam database. Pengujian ini menunjukkan keberhasilan dalam mengimplementasikan arsitektur microservice Cloud Firestore pada backend dari aplikasi pendeteksi kepiting soka.</td></tr>
<tr><td>6.</td><td>Apakah arsitektur Cloud Storage Bucket yang di implementasikan melalui Flask API dapat menyimpan gambar hasil klasifikasi?</td><td>Arsitektur Cloud Storage Bucket yang di implementasikan melalui Flask API dapat menyimpan gambar hasil klasifikasi.</td></tr>
<tr><td></td><td colspan="2"> 

Gambar 4.19 Hasil Pengujian Simpan Gambar

Gambar 4.19 memberikan visualisasi tentang keberhasilan pengujian dalam menjalankan fungsi simpan gambar melalui Flask API ke Cloud Storage Bucket. Hasil positif dari pengujian ini menunjukkan bahwa implementasi fungsi simpan gambar pada aplikasi berjalan sesuai harapan dan memberikan respons yang diinginkan. Kesuksesan pengujian ini menunjukkan bahwa API dapat beroperasi secara optimal dan memberikan respons yang memadai dalam mengirim data atau informasi yang diperlukan.</td></tr>
<tr><td>7.</td><td>Apakah arsitektur microservice yang di implementasikan melalui Flask API dapat menjalankan metode GET untuk</td><td>Arsitektur microservice yang di implementasikan melalui Flask API dapat menjalankan metode GET untuk</td></tr>
</table>

No.	Kasus Pengujian	Hasil Pengujian
	mendapatkan riwayat klasifikasi?	mendapatkan riwayat klasifikasi.



Gambar 4.20 Hasil Pengujian Metode *GET*

Gambar 4.20 memberikan visualisasi tentang keberhasilan pengujian dalam menjalankan fungsi *GET* melalui *platform* Postman. Hasil positif dari pengujian ini menunjukkan bahwa implementasi fungsi *GET* pada aplikasi berjalan sesuai harapan dan memberikan respons yang diinginkan.

4.4.5. Analisis Hasil Pengujian

Hasil pengujian API yang terdokumentasi dalam kasus-kasus pengujian di atas memberikan gambaran menyeluruh tentang kinerja dan responsivitas API dalam berbagai skenario.

a. Pada kasus pengujian kecepatan respons, API terbukti mampu memberikan respons yang sesuai dan cepat saat menerima permintaan dari *client*, sebagaimana terlihat pada Gambar 4.14. Keberhasilan ini memberikan keyakinan bahwa API siap digunakan dalam lingkungan produksi, dengan kemampuan merespons permintaan pengguna dengan efisien dalam beberapa detik.

b. Pada kasus pengujian memproses model *machine learning* untuk klasifikasi gambar, hasil pengujian pada Gambar 4.15 menunjukkan keberhasilan yang signifikan. API dapat memproses model *machine learning* dengan baik, memberikan hasil klasifikasi gambar yang akurat dan informatif dalam format

JSON. Tingkat kepercayaan yang tinggi, mencapai 93%, memberikan indikasi bahwa model mampu mengidentifikasi dan mengklasifikasikan gambar dengan tingkat akurasi yang tinggi.

c. Pada kasus pengujian respons gagal jika format gambar tidak sesuai, Gambar 4.16 menyoroti keberhasilan API dalam memberikan respons *error* ketika model mendeteksi format gambar yang tidak sesuai dengan ketentuan. Respons ini penting untuk memberikan petunjuk yang jelas kepada pengguna terkait masalah format yang perlu diperbaiki.
d. Pada kasus pengujian respons gagal jika gambar tidak termasuk dalam kelas klasifikasi model, Gambar 4.17 juga memberikan hasil yang baik. API dapat memberikan respons *error* ketika model tidak dapat mengenali gambar sebagai bagian dari salah satu kelas klasifikasinya. Analisis lebih lanjut terhadap respons JSON mengidentifikasi karakteristik gambar yang mungkin menjadi penyebab kegagalan, memberikan informasi berharga untuk perbaikan model ke depan.
e. Pada kasus pengujian terhadap arsitektur *Cloud Firestore* menunjukkan bahwa aplikasi berhasil menyimpan hasil klasifikasi dalam *database* yang ditunjukkan pada Gambar 4.18
f. Pada kasus pengujian fungsi simpan gambar melalui Flask API ke *Cloud Storage Bucket*, Gambar 4.19 menunjukkan bahwa implementasi fungsi simpan gambar pada aplikasi berjalan sesuai harapan dan memberikan respons yang diinginkan.
g. Pada Gambar 4.20 telah menunjukkan keberhasilan pengujian dalam menjalankan fungsi *GET* melalui *platform* Postman. Hasil positif dari pengujian ini menunjukkan bahwa implementasi fungsi *GET* pada aplikasi berjalan sesuai harapan dan memberikan respons yang diinginkan.

Secara keseluruhan, hasil pengujian ini memberikan pemahaman mendalam tentang kemampuan dan keterbatasan API dalam menghadapi berbagai skenario penggunaan. Dengan melibatkan Postman sebagai alat uji coba, dapat secara efektif mengeksplorasi berbagai kondisi dan memastikan bahwa API yang dibangun dengan arsitektur *microservice* siap untuk diimplementasikan dengan

kualitas dan keandalan tinggi. Analisis hasil ini menjadi landasan untuk perbaikan, pemeliharaan, dan peningkatan kinerja API ke depannya.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pembahasan yang telah dijabarkan pada bab sebelumnya, maka dapat disimpulkan dalam beberapa poin berikut:

1. Implementasi Arsitektur *Microservice* pada Aplikasi Pendeteksi Kepiting Soka telah berhasil dilakukan dengan baik yang dibuktikan dengan pengujian API melalui Postman.
2. Selama melakukan Implementasi Arsitektur *Microservice* pada Aplikasi Pendeteksi Kepiting Soka telah mengikuti metodologi pengembangan *waterfall* yang digunakan dengan baik

5.2. Saran

Beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut dari Implementasi Arsitektur *Microservice* pada Aplikasi Pendeteksi Kepiting Soka adalah:

1. Melakukan analisis lebih lanjut terkait efisiensi dan kinerja arsitektur *microservices*, termasuk pemantauan performa, manajemen beban, dan peningkatan skalabilitas jika diperlukan.
2. Menerapkan strategi keamanan tambahan pada setiap layanan *microservices* untuk memastikan keamanan data dan sistem secara menyeluruh.
3. Melibatkan pengguna dalam fase uji coba lebih lanjut untuk mendapatkan umpan balik langsung terkait pengalaman pengguna dan bagian-bagian yang perlu ditingkatkan.

DAFTAR PUSTAKA

[1] Jefri, "Kajian Kualitas Air Pada Budidaya Kepiting Soka (Scylla sp) di Pulau Tarakan," Universitas Borneo Tarakan, 2016.

[2] A. Romadhon, E. Prasetiyono, and A. M. Farhaby, "Laju Pertumbuhan dan Kecepatan Molting Kepiting Bakau (Scylla spp.) Dengan Pemberian Ekstrak Daun Pakis Hutan (Diplazium caudatum)," *J. Trop. Mar. Sci.*, vol. 5, no. April, pp. 9–18, 2022.

[3] M. Masitah, D. Rukmana, and B. Budimawan, "Analisis Produksi Kepiting Bakau (Scylla seratta) Kabupaten Bone," *J. Agribisnis Lahan Kering*, vol. 4, no. 4, pp. 49–52, 2019, doi: 10.32938/ag.v4i3.817.

[4] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3152803.

[5] Kemendikbud, "Sejarah Kementrian Pendidikan dan Kebudayaan," *Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi*, 2015. https://kemdikbud.go.id/main/tentang-kemdikbud/sejarah-kemdikbudristek (accessed Jan. 30, 2024).

[6] "Sejarah Singkat Kemendikbudristek," *INSPEKTORAT JENDERAL KEMENDIKBUDRISTEK*, 2023. https://itjen.kemdikbud.go.id/web/sejarah-singkat-kemendikbudristek/ (accessed Jan. 30, 2024).

[7] Kemendikbud, "Logo Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi," *Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi*, 2015. https://www.kemdikbud.go.id/main/informasi-publik/logo-kemdikbudristek (accessed Jan. 30, 2024).

[8] Sekretariat, "Sejarah Merdeka Belajar - Kampus Merdeka (MBKM)," *Kampus Merdeka MBKM USM*, 2021. https://mbkm.usm.ac.id/sejarah/ (accessed Jan. 30, 2024).

[9] LLDIKTI, "Logo Kampus Merdeka Indonesia Jaya," *Lembaga Layanan Pendidikan Tinggi Wilayah X*, 2020. https://lldikti10.id/detail/logo-kampus-merdeka-indonesia-jaya (accessed Jan. 30, 2024).

[10] R. F. Falah, "PERANCANGAN MICROSERVICE BERBASIS REST API PADA GOOGLE CLOUD PLATFORM MENGGUNAKAN NODEJS DAN PYTHON (STUDI KASUS: APLIKASI PENDETEKSI PENYAKIT DAUN PADI)," Lampung, 2023. [Online]. Available: https://digilib.unila.ac.id/74644/3/3. SKRIPSI TANPA PEMBAHASAN.pdf.

[11] L. S. Hidayat, "Laporan Magang Studi Independen Bersertifikat Android Learning Path Di Bangkit Academy 2023 By Google, Goto, Traveloka Di Yayasan Dicoding Indonesia," Ponogoro, 2023. [Online]. Available: https://sikap.unida.gontor.ac.id/simak-magang/download?id=f2fb7031-1d1e-4396-ad26-0afa9729694a.

[12] F. B. W. Almaliki, "Optimalisasi Dosis Pemberian Ekstrak Daun Karamunting (Melastoma malabathricum L) Pada Proses Ganti Kulit Kepiting Bakau (Scylla serrata)," Universitas Borneo Tarakan, 2021.

[13] I. Shukri, "Menghasilkan Kepiting Soka Berkualitas dengan Budidaya yang

Tepat," *Trubus.id*, 2023. https://trubus.id/menghasilkan-kepiting-soka-berkualitas-dengan-budidaya-yang-tepat/ (accessed Jan. 13, 2023).
[14] M. Haikal, N. Rahmadina, S. Berliani, and A. Kurniawan, "Model Budidaya Kepiting Soka Skala Rumah Tangga Sistem Apartemen Sebagai Sarana Edukasi Masyarakat Pulau Bangka," *Literasi J. Pengabdi. Masy. dan Inov.*, vol. 2, no. 1, pp. 8–14, 2022, doi: 10.58466/literasi.v2i1.155.
[15] A. Nurpadillah, Tyara, and Rijaldi, "Vertical Crab House." IFMOS Institut Pertanian Bogor, Bogor, 2023.
[16] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 1st ed. Ottawa, Canada: Apress Berkeley, CA, 2019.
[17] A. B. Wicaksono, R. Munadi, and Sussi, "Cloud Server Design for Heavy Workload Gaming Computing with Google Cloud Platform," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 2, 2032, doi: 10.11591/ijece.v13i2.pp2197-2205.
[18] V. Lakshmanan, *Data Science on the Google Cloud Platform*, 2nd ed. O'Reilly Media, 2022.
[19] W. T. Sung, I. G. T. Isa, and S. J. Hsiao, "An IoT-Based Aquaculture Monitoring System Using Firebase," *Comput. Mater. Contin.*, vol. 76, no. 2, pp. 2180–2200, 2023, doi: 10.32604/cmc.2023.041022.
[20] B. Pang, E. Nijkamp, and Ying Nian Wu, "Deep Learning With TensorFlow: A Review," *J. Educ. Behav. Stat.*, vol. 45, no. 2, pp. 227–248, 2019, doi: https://doi.org/10.3102/1076998619872761.
[21] J. Dai, "Real-time and accurate object detection on edge device with TensorFlow Lite," *J. Phys. Conf. Ser.*, pp. 21–23, 2020, doi: 10.1088/1742-6596/1651/1/012114.
[22] L. Ramalho, *Fluent Python*, 2nd ed. California: O'Reilly Media, 2022.
[23] D. A. Anggoro and N. C. Aziz, "Implementation of K-Nearest Neighbors Algorithm for Predicting Heart Disease Using Python Flask," *Iraqi J. Sci.*, vol. 62, no. 9, pp. 3196–3219, 2021, doi: 10.24996/ijs.2021.62.9.33.
[24] A. Z. Ikromovna, "Programming Environments for Creating Mobile Applications on the Android Operating System," *Am. J. Public Dipl. Int. Stud.*, vol. 01, no. 10, pp. 305–309, 2023.
[25] D. Westerveld, *API Testing and Development with Postman*, 1st ed. Birmingham: Packt Publishing Ltd., 2021.
[26] H. Kurniawan, W. Apriliah, I. Kurnia, and D. Firmansyah, "Penerapan Metode Waterfall Dalam Perancangan Sistem Informasi Penggajian Pada Smk Bina Karya Karawang," *J. Interkom J. Publ. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 14, no. 4, pp. 13–23, 2021, doi: 10.35969/interkom.v14i4.78.
[27] AdminLP2M, "Metode Waterfall – Definisi dan Tahap-tahap Pelaksanaannya," *Lembaga Penelitian dan Pengabdian Masyarakat Universitas Medan Area*, 2022. https://lp2m.uma.ac.id/2022/06/07/metode-waterfall-definisi-dan-tahap-tahap-pelaksanaannya/ (accessed Jan. 14, 2024).
[28] R. S. Pressman, "Waterfall Model," in *Software Engineering: A Practiotioner's Approach*, 8th ed., New York: McGraw-Hill Education, 2015, pp. 48–54.
[29] A. L. Kalua, Veronika H, and D. T. Salaki, "Sistem Pakar Diagnosa Penyakit Malaria dengan Certainty Factor dan Forward Chaining," *J. Inf.*

Technol. Softw. Eng. Comput. Sci., vol. 1, no. 1, pp. 22–34, 2022, doi: 10.58602/itsecs.v1i1.10.

[30] Silvia, "Pengertian Microservices, Karakteristik, dan Kelebihannya," *Jetorbit*, 2022. https://www.jetorbit.com/blog/pengertian-microservices-dan-kelebihannya/ (accessed Jan. 31, 2024).