

Turnitin Dini

by Dini R

Submission date: 31-Jul-2026 10:02PM (UTC+0700)

Submission ID: 3006121383

File name: in_Final_Skripsi_2210511093_Dini_Rahmawati_-_Dini_Rahmawati.docx (20.96M)

Word count: 29760

Character count: 207112

PERANCANGAN EKSTENSI CHROME BERBASIS *FIREFLY ALGORITHM* UNTUK OPTIMASI *SCRAPING* DAN *IMAGE TEXT SWAPPING* PADA PLATFORM 1688

Dini Rahmawati

ABSTRAK

Pelaku usaha impor di Indonesia banyak memanfaatkan platform 1688.com sebagai sumber produk. Namun, informasi penting yang tertanam pada gambar produk (*in-image text*) disajikan dalam bahasa Mandarin sehingga sulit dipahami menggunakan penerjemah konvensional. Penelitian ini bertujuan merancang ekstensi Chrome yang mampu melakukan *web scraping* data produk secara otomatis, menerapkan *Image Text Swapping* berbasis *Optical Character Recognition* (OCR), serta mengoptimalkan *Pipeline Scheduler* menggunakan *Firefly Algorithm*. Penelitian ini menggunakan metode *Research and Development* (R&D) dengan model pengembangan perangkat lunak *V-Model*. Evaluasi sistem dilakukan melalui pengujian fungsional pada setiap modul, pengujian optimasi *Pipeline Scheduler* menggunakan *Firefly Algorithm*, serta *User Acceptance Testing* (UAT) untuk mengukur tingkat penerimaan pengguna. Hasil pengujian menunjukkan bahwa modul OCR memperoleh tingkat akurasi sebesar 86,41%, sedangkan modul *Translation* berhasil memproses seluruh permintaan translasi dengan *Success Rate* 100%. Penerapan *Firefly Algorithm* mampu mengoptimalkan konfigurasi *Pipeline Scheduler* secara adaptif sehingga waktu pemrosesan satu halaman produk berkurang dari 202,140 detik menjadi 20,63 detik, atau mengalami pengurangan waktu sebesar 89,79% dibandingkan *Sequential Processing*. Selain itu, hasil *User Acceptance Testing* (UAT) memperoleh tingkat penerimaan pengguna sebesar 84,38% dengan kategori Sangat Baik. Hasil penelitian menunjukkan bahwa sistem mampu membantu proses pengambilan data produk, penerjemahan gambar, dan penyusunan katalog produk multibahasa secara lebih cepat, akurat, dan efisien.

Kata Kunci: *Chrome Extension, Firefly Algorithm, Image Text Swapping, OCR, Web Scraping*

DESIGN OF A FIREFLY ALGORITHM-BASED CHROME EXTENSION FOR OPTIMIZING SCRAPING AND IMAGE TEXT SWAPPING ON THE 1688 PLATFORM

Dini Rahmawati

ABSTRACT

Import businesses in Indonesia widely use the 1688 platform as a source of products. However, important information embedded in product images (in-image text) is presented in Mandarin, making it difficult to understand using conventional translation tools. This research aims to design a Chrome Extension capable of automatically performing product data web scraping, implementing Optical Character Recognition (OCR)-based Image Text Swapping⁵⁹ and optimizing the Pipeline Scheduler using the Firefly Algorithm. This research employed the Research and Development (R&D) method with the V-Model software development model. System evaluation was conducted through functional tests¹⁰⁴ of each module, Pipeline Scheduler optimization testing using the Firefly Algorithm, and User Acceptance Testing (UAT). The results show that the OCR module achieved an accuracy of 86.41%, while the Translation module achieved a 100% success rate. The implementation of the Firefly Algorithm reduced the processing time of a single product page from 202.140 seconds to 20.63 seconds, representing an 89.79% reduction⁴⁴ compared to Sequential Processing. Furthermore, the UAT result reached 84.38%, categorized as Very Good. These findings demonstrate that the proposed system effectively supports product data extraction, image translation, and multilingual product catalog generation with improved speed, accuracy, and efficiency.

Keywords: Chrome Extension, Firefly Algorithm, Image Text Swapping, OCR, Web Scraping

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi digital telah mendorong pertumbuhan pesat sektor e-commerce global. Berdasarkan laporan Statista (2024), nilai pasar e-commerce dunia diproyeksikan mencapai lebih dari 6,3 triliun USD pada tahun 2025, dengan peningkatan signifikan pada aktivitas perdagangan lintas negara, termasuk impor produk melalui platform daring. Kondisi ini membuka peluang besar bagi pelaku usaha impor di Indonesia untuk memperoleh produk langsung dari produsen luar negeri.

Salah satu platform yang banyak dimanfaatkan oleh pelaku usaha impor Indonesia adalah 1688.com, sebuah platform Business to Business (B2B) asal Tiongkok yang menyediakan jutaan data produk dari berbagai produsen. Penggunaan platform ini dapat diidentifikasi dari kemiripan foto katalog produk yang dijual di marketplace lokal Indonesia seperti Shopee dan Tokopedia dengan gambar yang terdapat pada halaman produk di 1688.com. Hal ini mengindikasikan bahwa sebagian besar pelaku usaha impor Indonesia menggunakan 1688.com sebagai sumber produk utama dan secara langsung memanfaatkan aset visual dari platform tersebut untuk keperluan penjualan lokal.

Meskipun menawarkan harga kompetitif dan variasi produk yang luas, pemanfaatan data produk dari 1688.com masih menghadapi kendala utama, yaitu seluruh antarmuka dan informasi produk disajikan dalam Bahasa Mandarin. Tantangan terbesar terletak pada konten visual produk, di mana spesifikasi teknis, fitur keunggulan, dan informasi penting lainnya sering disajikan dalam bentuk teks yang tertanam langsung di dalam gambar produk (in-image text). Teks yang tertanam dalam gambar tersebut tidak dapat dikenali atau diterjemahkan menggunakan ekstensi penerjemah konvensional seperti Google Translate Extension, karena ekstensi tersebut hanya bekerja pada teks HTML yang dapat dibaca secara struktural, bukan pada teks yang menjadi bagian dari elemen visual gambar.

Hambatan bahasa pada gambar produk ini secara langsung berdampak pada proses analisis produk (product analysis) yang menjadi tahap krusial sebelum pelaku usaha impor mengambil keputusan pembelian. Dalam praktik pengadaan produk impor, pelaku usaha perlu memahami secara menyeluruh keunggulan, spesifikasi teknis, dan diferensiasi produk yang umumnya disajikan dalam bentuk infografis pada gambar katalog 1688.com. Ketidakmampuan membaca informasi tersebut secara langsung menyebabkan pelaku usaha impor tidak dapat melakukan evaluasi kualitas produk secara efisien, sehingga berisiko mengambil keputusan pengadaan yang kurang tepat. Kondisi ini menjadi justifikasi utama kebutuhan akan sistem yang mampu menerjemahkan teks pada gambar produk secara otomatis dan

langsung di halaman 1688.com, sehingga proses product analysis dapat dilakukan dengan lebih cepat, akurat, dan efisien.

Akibat keterbatasan tersebut, pelaku usaha impor di Indonesia masih melakukan proses pencarian dan pemahaman informasi produk secara manual. Proses ini umumnya dilakukan dengan membuka halaman produk pada platform 1688.com satu per satu, kemudian mengunduh gambar produk dan menerjemahkan teks berbahasa Mandarin yang terdapat di dalam gambar menggunakan layanan penerjemah gambar yang hanya mampu memproses satu gambar dalam satu waktu. Proses yang berulang tersebut menyebabkan pelaku usaha membutuhkan waktu yang cukup lama untuk memahami spesifikasi dan informasi produk, terutama ketika harus menganalisis puluhan hingga ratusan produk sekaligus. Selain itu, konten pada platform 1688.com dimuat secara dinamis menggunakan teknologi Asynchronous JavaScript and XML (AJAX) serta arsitektur Single Page Application (SPA), sehingga metode pengambilan data berbasis HTTP request konvensional tidak efektif diterapkan pada platform tersebut.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan perancangan ekstensi Chrome untuk melakukan web scraping data produk dari platform 1688.com yang dilengkapi dengan Optical Character Recognition (OCR) dan mekanisme image text swapping. Sistem dirancang untuk mengambil informasi produk secara otomatis, mengenali teks berbahasa Mandarin yang terdapat pada gambar produk menggunakan OCR, kemudian menerjemahkannya ke dalam bahasa Indonesia dan mengganti teks hasil terjemahan secara langsung pada tampilan gambar di halaman 1688.com. Dengan demikian, pelaku usaha impor dapat memahami informasi produk secara langsung tanpa harus berpindah ke aplikasi penerjemah maupun mengunduh gambar terlebih dahulu, sehingga proses analisis dan evaluasi produk menjadi lebih efisien sebelum keputusan pembelian dilakukan.

Untuk meningkatkan efisiensi proses scraping dan image text swapping, penelitian ini menerapkan Firefly Algorithm (FA) sebagai mekanisme optimasi pada pipeline scheduler. Algoritma ini digunakan untuk menentukan konfigurasi jumlah tugas yang dapat dijalankan secara bersamaan (*concurrency limit*) pada setiap tahapan pemrosesan, yang meliputi ekstraksi URL gambar dari *Document Object Model (DOM)* halaman 1688, pengunduhan gambar, *Optical Character Recognition (OCR)*, penerjemahan teks, dan *image text swapping*. Selama proses berlangsung, konfigurasi *scheduler* dapat disesuaikan secara adaptif berdasarkan kondisi performa sistem. Setiap konfigurasi dievaluasi menggunakan metrik kinerja seperti waktu penyelesaian proses (*makespan*), *throughput*, dan keseimbangan beban antar tahapan (*pipeline balance*), sehingga diperoleh konfigurasi scheduler yang mampu meningkatkan efisiensi proses scraping dan pemrosesan gambar. Selain itu, sistem juga mengintegrasikan Gemini API untuk menghasilkan

ringkasan deskripsi produk berbahasa Indonesia secara otomatis berdasarkan hasil ekstraksi OCR. Integrasi antara *web scraping*, *OCR*, *image text swapping*, *Firefly Algorithm*, dan kecerdasan buatan diharapkan mampu membantu pelaku usaha impor memahami informasi produk pada platform 1688.com secara lebih cepat, akurat, dan efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang ekstensi Chrome yang mampu melakukan v data foto produk dari platform 1688 secara otomatis?
2. Bagaimana mengimplementasikan image text swapping berbasis OCR untuk melakukan penerjemahan dan penggantian teks berbahasa Mandarin pada gambar produk secara langsung di halaman website 1688?
3. Bagaimana menerapkan Firefly Algorithm untuk mengoptimalkan penjadwalan proses scraping dan image text swapping pada sistem yang dikembangkan?

1.3 Batasan Masalah

Agar penelitian lebih terarah, batasan masalah ditetapkan sebagai berikut:

1. Platform yang digunakan sebagai sumber data adalah 1688.com.
2. Proses scraping dibatasi pada pengambilan informasi produk yang meliputi nama produk, harga, gambar produk, informasi toko, URL produk, serta data pendukung yang tersedia pada halaman produk.
3. OCR hanya diterapkan pada gambar produk yang memiliki teks berbahasa Mandarin atau Inggris.
4. Penerjemahan teks dilakukan menggunakan Google Translate API dengan bahasa Indonesia sebagai bahasa utama keluaran sistem.
5. Firefly Algorithm hanya diterapkan sebagai mekanisme optimasi Pipeline Scheduler dan tidak digunakan untuk meningkatkan akurasi OCR, hasil translasi, maupun kualitas image text swapping.
6. Sistem dikembangkan dalam bentuk Chrome Extension berbasis Manifest V3 serta Web Dashboard.
7. Sistem hanya dapat dijalankan pada browser Google Chrome yang mendukung Chrome Extension Manifest V3.
8. Aspek keamanan dan autentikasi akun pengguna tidak menjadi fokus utama penelitian.

9. Web Dashboard hanya digunakan untuk menampilkan, mencari, dan mengelola hasil scraping serta hasil pengolahan gambar.
10. OCR hanya diterapkan pada gambar produk dan tidak mencakup teks yang berada di luar area gambar.
11. Gemini AI hanya digunakan untuk menghasilkan ringkasan deskripsi produk dan tidak digunakan pada proses scraping, OCR, maupun penerjemahan gambar.
12. Pengambilan data dilakukan menggunakan DOM Scraping terhadap halaman yang telah selesai di render oleh browser.
13. User Acceptance Testing (UAT) dibatasi pada evaluasi penerimaan pengguna terhadap fitur-fitur utama sistem dan tidak mencakup pengujian keamanan, kompatibilitas lintas browser, maupun performa jangka panjang.

62 1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan membangun ekstensi Chrome yang mampu melakukan web scraping data produk dari platform 1688 secara otomatis.
2. Mengimplementasikan fitur image text swapping berbasis OCR untuk melakukan penerjemahan dan penggantian teks Mandarin pada gambar produk secara langsung di website 1688.
3. Menerapkan Firefly Algorithm untuk mengoptimalkan penjadwalan proses scraping dan image text swapping pada sistem yang dikembangkan.

111 1.5 Manfaat Penelitian

Bagi Peneliti

1. Meningkatkan pemahaman dalam penerapan Firefly Algorithm sebagai mekanisme optimasi penjadwalan (*scheduler*) pada sistem berbasis Chrome Extension yang terintegrasi dengan teknologi OCR dan kecerdasan buatan.
2. Menjadi pengalaman praktis dalam pengembangan perangkat lunak berbasis Chrome Extension yang mengintegrasikan Optical Character Recognition (OCR), Firefly Algorithm, Image Text Swapping, serta Large Language Model (LLM) Gemini.

Bagi Akademisi

1. Menjadi referensi bagi penelitian selanjutnya mengenai optimasi proses web scraping, OCR multibahasa, serta pengembangan sistem berbasis Chrome Extension yang terintegrasi dengan kecerdasan buatan.
2. Memberikan kontribusi bagi pengembangan penelitian di bidang rekayasa perangkat lunak (*software engineering*), kecerdasan buatan (*artificial intelligence*), dan pengolahan citra digital (*image processing*).

Bagi Praktisi dan Industri

1. Membantu pelaku usaha atau importir dalam mempercepat proses riset penerjemahan dan pembuatan kebutuhan katalog produk multibahasa.

1.6 Waktu dan Tempat Penelitian

1.6.1 Waktu Penelitian

Penelitian ini akan dilaksanakan selama 10 bulan yaitu dari bulan September 2025 sampai bulan Juni 2026.

1.6.2 Tempat Penelitian

Penulis akan melakukan penelitian pada Fakultas Ilmu Komputer UPN “Veteran” Jakarta yang terletak di Jalan Rumah Sakit Fatmawati Raya, Pondok Labu, Cilandak, Jakarta Selatan, Jakarta.

1.7 Sistematika Penulisan

Penulisan laporan penelitian ini disusun dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, waktu, tempat, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Berisi teori-teori terkait seperti Web Scraping, OCR, Firefly Algorithm, Chrome Extension Development, serta penelitian terdahulu yang relevan.

BAB III METODOLOGI PENELITIAN

Menjelaskan tahapan penelitian, perancangan sistem, arsitektur teknis, algoritma yang digunakan, serta alat dan bahan penelitian.

BAB IV HASIL DAN PEMBAHASAN

Menampilkan implementasi sistem, hasil pengujian, serta analisis performa dan efektivitas optimasi.

100 BAB

V

PENUTUP

Menyajikan kesimpulan dan saran pengembangan lebih lanjut.

BAB 2. TINJAUAN PUSTAKA

2.1 Web Scraping

Web scraping adalah proses otomatisasi pengumpulan, ekstraksi, dan penyimpanan data dari halaman web menggunakan perangkat lunak atau skrip pemrograman (Zhao, 2017). Secara umum, *web scraping* bekerja dengan mengirimkan permintaan HTTP ke *server* target dan menganalisis respon HTML yang diterima. Namun, pada platform *e-commerce* modern yang memanfaatkan pemuatan konten secara dinamis (*dynamic client-side rendering*), pendekatan ekstraksi kode sumber HTML mentah (*raw HTML*) sering kali mengalami kendala karena data produk belum sepenuhnya dimuat saat *request* awal dilakukan (Chandra & Tiwari, 2021).

Untuk mengatasi keterbatasan tersebut, teknik *web scraping* berkembang menjadi *DOM-based Scraping* yang berfokus pada pembacaan struktur elemen *Document Object Model* (DOM) di sisi *browser client* setelah seluruh skrip JavaScript selesai dieksekusi. Dalam penelitian ini, *web scraping* diterapkan melalui komponen *Content Script* pada ekstensi peramban untuk mengekstraksi data produk secara *real-time* langsung dari antarmuka visual halaman web 1688.com.

99

2.2 Document Object Model (DOM)

Document Object Model (DOM) adalah antarmuka pemrograman aplikasi (*Application Programming Interface / API*) standar yang mendefinisikan struktur dokumen HTML dan XML sebagai struktur pohon (*DOM Tree*) berhierarki (W3C, 2020). Setiap elemen, atribut, dan teks di dalam dokumen dipetakan sebagai node/objek yang dapat diakses, dimanipulasi, dan diperbarui secara dinamis menggunakan bahasa pemrograman skrip seperti JavaScript (Flanagan, 2020).

129

Pada platform *e-commerce* seperti 1688.com yang menggunakan pemuatan konten dinamis berbasis AJAX dan Single Page Application (SPA), struktur HTML mentah yang dikirim oleh server sering kali belum memuat data produk secara utuh (Chandra & Tiwari, 2021). Melalui manipulasi dan pembacaan DOM setelah proses *rendering* selesai pada *browser client*, ekstensi peramban dapat mengekstraksi informasi produk (seperti judul, harga, variasi, dan URL gambar) secara akurat sesuai dengan elemen visual yang sedang ditampilkan kepada pengguna (Varghese & Vijay, 2019).

128

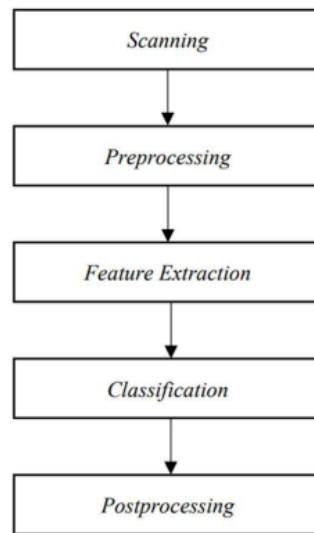
2.3 Optical Character Recognition (OCR)

Teknologi Optical Character Recognition (OCR) berfungsi mengekstrak teks dari gambar, memungkinkan sistem membaca informasi visual secara

otomatis. OCR menjadi elemen penting dalam penelitian ini karena banyak produk 1688 yang menampilkan informasi dalam bentuk teks tertanam di gambar.

OCR modern seperti Tesseract 5.0 atau EasyOCR telah mendukung berbagai bahasa, termasuk Mandarin. Namun, akurasi pengenalan dipengaruhi oleh kualitas gambar, pencahayaan, dan jenis font (Chen et al., 2021).

Akurasi OCR sangatlah krusial dan dipengaruhi oleh kualitas pra-pemrosesan citra, seperti *binarization* dan *noise reduction* (Gabor et al., 2018). Lebih lanjut, karakter Mandarin dikenal memiliki tingkat kesulitan pengenalan yang tinggi. Wang et al. (2020) dalam studi *Hybrid model for Chinese character recognition based on Tesseract-OCR* menunjukkan bahwa akurasi pengenalan karakter Tionghoa dapat ditingkatkan hingga sekitar 12% melalui kombinasi pra-pemrosesan citra yang spesifik dan penggunaan model koreksi pasca-OCR (*post-OCR correction*). Penelitian ini menegaskan pentingnya optimasi tahapan awal (pra-pemrosesan) untuk meningkatkan akurasi ekstraksi teks Mandarin, menjadikannya target utama optimasi dalam penelitian ini.



Gambar 2.1 Proses OCR

2.3.1 EasyOCR Framework

EasyOCR merupakan salah satu pustaka (*library*) OCR terbuka sumber (*open-source*) berbasis Python yang dikembangkan oleh Jaied AI. EasyOCR dibangun di atas kerangka kerja pembelajaran dalam (*Deep Learning*) PyTorch dan

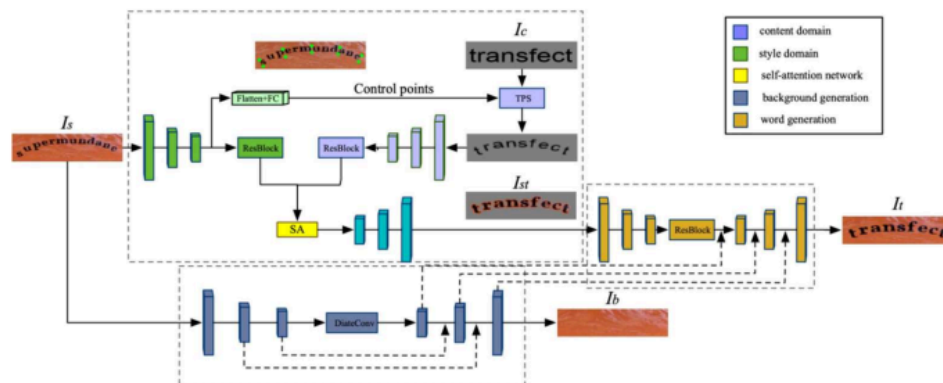
dirancang untuk mendukung pengenalan teks multibahasa, termasuk teks berbahasa Mandarin dan Inggris (Jaided AI, 2020).

Dalam arsitekturnya, EasyOCR mengintegrasikan dua model utama, yaitu CRAFT (*Character Region Awareness for Text Detection*) yang berfungsi untuk mendeteksi area/lokasi batas teks (*bounding box*) pada citra, serta CRNN (*Convolutional Recurrent Neural Network*) yang digabungkan dengan CTC Loss (*Connectionist Temporal Classification*) untuk mengenali dan memprediksi urutan karakter di dalam area teks tersebut. Keunggulan arsitektur ini menjadikan EasyOCR responsif dan akurat dalam mengekstraksi teks dari gambar produk sebelum diteruskan ke tahap penerjemahan dan *image text swapping*.

2.4 Image Text Swapping

Image Text Swapping, yang juga dikenal sebagai *scene text editing*, merupakan teknik pengolahan citra yang bertujuan untuk menghapus teks asli pada suatu gambar dan menggantinya dengan teks baru tanpa merusak struktur visual gambar tersebut. Teknik ini memungkinkan penggantian teks dilakukan dengan tetap mempertahankan konteks visual, seperti warna latar, tekstur, dan tata letak objek di sekitarnya. Menurut Zhang et al. (2021), image text swapping banyak digunakan dalam sistem visual multibahasa untuk meningkatkan keterbacaan informasi pada gambar lintas bahasa.

Setelah proses ekstraksi dan translasi teks (OCR) selesai, tahapan selanjutnya adalah Image Text Swapping atau *Scene Text Editing*. Proses ini adalah inti dari solusi visual multibahasa yang diusulkan. *Image Text Swapping* melibatkan dua langkah kompleks: (1) Inpainting, yaitu penghapusan teks sumber dari gambar, dan (2) Text Rendering, yaitu penyisipan teks terjemahan ke lokasi yang sama dengan tetap mempertahankan atribut visual (*font, warna, background*). Yang et al. (2020) dalam paper SwapText: Image Based Texts Transfer in Scenes mengusulkan kerangka yang efektif untuk mentransfer teks antar gambar dengan preservasi tekstur latar belakang.



155

Gambar 2.2 Arsitektur Model SwapText (Yang et al., 2020)

Penelitian oleh Yang et al. (2020) melalui model *SwapText* menunjukkan bahwa pendekatan berbasis inpainting dan text rendering mampu menghasilkan penggantian teks yang lebih natural dengan preservasi tekstur latar belakang.

Dalam konteks penelitian ini, proses *image text swapping* akan disesuaikan menggunakan pendekatan OCR (Optical Character Recognition) untuk mengekstrak teks Mandarin dari gambar produk, kemudian menerjemahkannya menggunakan model NLP multibahasa seperti MarianNMT atau Gemini API, dan akhirnya mengganti teks lama dengan teks hasil translasi menggunakan kombinasi OpenCV dan Pillow.

2.5 Firefly Algorithm

2.5.1 Pengertian Firefly Algorithm

Firefly Algorithm (FA) merupakan salah satu algoritma optimasi berbasis populasi (*population-based metaheuristic*) yang diperkenalkan oleh Xin-She Yang pada tahun 2008. Algoritma ini terinspirasi dari perilaku alami kunang-kunang (*fireflies*) yang menggunakan intensitas cahaya sebagai mekanisme komunikasi untuk menarik individu lain. Dalam Firefly Algorithm, tingkat kecerahan (*brightness*) merepresentasikan kualitas solusi (*fitness*), sehingga individu dengan tingkat kecerahan yang lebih rendah akan bergerak menuju individu dengan tingkat kecerahan yang lebih tinggi (Yang, 2008).

Firefly Algorithm termasuk ke dalam kelompok algoritma *swarm intelligence* yang memanfaatkan interaksi antarindividu dalam suatu populasi untuk memperoleh solusi optimal. Algoritma ini memiliki kemampuan yang baik dalam menyeimbangkan proses eksplorasi (*exploration*) dan eksploitasi (*exploitation*), sehingga banyak diterapkan pada berbagai permasalahan optimasi, seperti optimasi parameter, penjadwalan (*scheduling*), alokasi sumber daya, serta optimasi jaringan (Yang, 2009).

Seiring perkembangannya, Firefly Algorithm terus mengalami pengembangan untuk menyelesaikan permasalahan optimasi yang lebih kompleks. Salah satu penerapannya adalah optimasi *workflow scheduling* pada lingkungan *hybrid cloud-edge* melalui modifikasi mekanisme Firefly untuk meningkatkan efisiensi penggunaan sumber daya, menurunkan *makespan*, serta meningkatkan *resource utilization*. Hasil penelitian tersebut menunjukkan bahwa Firefly Algorithm tetap relevan dan mampu memberikan performa yang baik pada permasalahan penjadwalan modern (Alsadie & Alsulami, 2024).

2.5.2 Pengertian Kerja Firefly Algorithm

Firefly Algorithm bekerja berdasarkan tiga aturan dasar yang mengadopsi perilaku alami kunang-kunang. Pertama, seluruh kunang-kunang dianggap tidak memiliki perbedaan jenis kelamin (*unisex*), sehingga setiap individu dapat saling menarik tanpa memperhatikan jenisnya. Kedua, tingkat ketertarikan (*attractiveness*) antar kunang-kunang berbanding lurus dengan intensitas cahaya (*brightness*) yang dipancarkan, sehingga kunang-kunang dengan tingkat kecerahan yang lebih rendah akan bergerak menuju kunang-kunang yang lebih terang. Ketiga, tingkat kecerahan setiap kunang-kunang ditentukan oleh nilai fungsi objektif (*fitness function*), sehingga solusi dengan nilai *fitness* yang lebih baik akan memiliki tingkat kecerahan yang lebih tinggi (Yang, 2009).

Pada Firefly Algorithm, jarak antara dua kunang-kunang dihitung menggunakan jarak Euclidean yang dinyatakan sebagai berikut (Yang, 2009).

$$r_{ij} = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2} \quad (2.1)$$

Keterangan:

- r_{ij} = jarak antara kunang-kunang ke- i dan ke- j
- $x_{i,k}$ = posisi kunang-kunang ke- i pada dimensi ke- k
- $x_{j,k}$ = posisi kunang-kunang ke- j pada dimensi ke- k
- d = jumlah dimensi pencarian

Semakin dekat jarak antar kunang-kunang, semakin besar pula daya tarik yang dihasilkan. Sebaliknya, daya tarik akan berkurang secara eksponensial seiring bertambahnya jarak antar individu. Hubungan tersebut dinyatakan melalui persamaan *attractiveness* berikut (Yang, 2009).

$$\beta(r) = \beta_0 e^{-\gamma r_{ij}^2} \quad (2.2)$$

Keterangan:

- $\beta(r)$ = tingkat ketertarikan pada jarak r
- β_0 = tingkat ketertarikan maksimum saat $r = 0$
- γ = koefisien penyerapan cahaya (*light absorption coefficient*)
- r_{ij} = jarak antar kunang-kunang

Pergerakan kunang-kunang menuju solusi yang lebih baik dilakukan dengan memperhatikan tingkat ketertarikan serta faktor acak (*randomization*) untuk

menjaga keseimbangan antara proses eksplorasi dan eksploitasi. Persamaan perpindahan posisi kunang-kunang dirumuskan sebagai berikut (Yang, 2009).

$$x_i = x_i + \beta 0 e^{\gamma r_{ij}^2} (x_j - x_i) + \alpha (rand - 0.50) \quad (2.3)$$

Keterangan:

- x_i = posisi kunang-kunang ke- i
- x_j = posisi kunang-kunang yang memiliki tingkat kecerahan lebih tinggi
- $\beta 0$ = tingkat ketertarikan maksimum
- γ = koefisien penyerapan cahaya
- α = parameter randomisasi
- $rand$ = bilangan acak dengan rentang 0 hingga 1

Setelah posisi kunang-kunang diperbarui, setiap solusi dievaluasi menggunakan fungsi objektif (*fitness function*) untuk menentukan tingkat kualitas solusi yang diperoleh. Nilai *fitness* digunakan sebagai dasar dalam menentukan tingkat kecerahan (*brightness*) setiap kunang-kunang sehingga solusi dengan nilai *fitness* yang lebih baik akan memiliki tingkat kecerahan yang lebih tinggi. Secara umum, fungsi *fitness* pada Firefly Algorithm dinyatakan sebagai berikut (Yang, 2009).

$$Fitness = f(x) \quad (2.4)$$

Keterangan:

Fitness = nilai kualitas solusi yang diperoleh.

$f(x)$ = fungsi objektif yang digunakan untuk mengevaluasi suatu solusi, dengan bentuk yang disesuaikan terhadap permasalahan optimasi yang diselesaikan.

Bentuk fungsi *fitness* pada Firefly Algorithm tidak bersifat tetap, melainkan disesuaikan dengan karakteristik permasalahan yang dioptimasi. Oleh karena itu, setiap penelitian dapat mendefinisikan fungsi *fitness* yang berbeda sesuai tujuan optimasi. Pada penelitian ini, fungsi *fitness* dirancang berdasarkan indikator throughput, parameter match, resource efficiency, dan retry rate. Perancangan fungsi *fitness* tersebut dijelaskan lebih lanjut pada Subbab 3.3.3.

Berdasarkan persamaan tersebut, proses optimasi pada Firefly Algorithm dilakukan secara iteratif. Setiap kunang-kunang terlebih dahulu dievaluasi menggunakan fungsi objektif untuk memperoleh tingkat kecerahan (*brightness*). Selanjutnya, jarak antar kunang-kunang dihitung menggunakan jarak Euclidean,

kemudian nilai daya tarik (attractiveness) ditentukan berdasarkan jarak tersebut. ² Kunang-kunang dengan tingkat kecerahan yang lebih rendah akan bergerak menuju kunang-kunang yang lebih terang menggunakan persamaan perpindahan. Proses ini dilakukan secara berulang hingga diperoleh solusi dengan nilai fungsi objektif terbaik.

Prinsip-prinsip tersebut memungkinkan Firefly Algorithm digunakan pada berbagai permasalahan optimasi, termasuk optimasi penjadwalan (*scheduling*) dan alokasi sumber daya. Kemampuan algoritma dalam menyeimbangkan eksplorasi dan eksploitasi menjadikannya sesuai untuk diterapkan pada sistem yang memerlukan penyesuaian konfigurasi secara adaptif terhadap perubahan kondisi lingkungan (Alsadie & Alsulami, 2024).

Firefly Algorithm dipilih pada penelitian ini karena memiliki kemampuan yang baik dalam menyelesaikan permasalahan optimasi yang melibatkan banyak kombinasi parameter tanpa memerlukan model matematis yang kompleks. Pada sistem yang dikembangkan, konfigurasi *Pipeline Scheduler* terdiri atas beberapa parameter, yaitu jumlah *Download Worker*, *OCR Worker*, *Translation Worker*, dan *Image Text Swapping Worker*. Kombinasi dari parameter-parameter tersebut membentuk ruang pencarian yang cukup besar sehingga diperlukan algoritma optimasi untuk memperoleh konfigurasi yang paling sesuai.

Dibandingkan dengan pendekatan *rule-based* yang menggunakan nilai parameter tetap, Firefly Algorithm mampu melakukan pencarian solusi secara adaptif berdasarkan kondisi aktual pipeline. Algoritma mengevaluasi setiap kandidat konfigurasi menggunakan fungsi *fitness* yang mempertimbangkan indikator performa seperti *throughput*, *parameter matching*, *resource efficiency*, dan *retry rate*. Dengan mekanisme tersebut, konfigurasi worker yang dihasilkan dapat menyesuaikan perubahan beban kerja sehingga distribusi proses pada setiap tahapan pipeline menjadi lebih seimbang.

Selain itu, Firefly Algorithm memiliki mekanisme keseimbangan antara proses eksplorasi (*exploration*) dan eksploitasi (*exploitation*) sehingga mampu mencari solusi yang baik tanpa terjebak pada solusi lokal. Karakteristik tersebut sesuai dengan kebutuhan penelitian ini yang memerlukan optimasi konfigurasi *Pipeline Scheduler* secara adaptif untuk meningkatkan efisiensi proses scraping dan *image text swapping* pada platform 1688.

2.5.3 Implementasi Firefly Algorithm pada Penelitian

Pada penelitian ini, Firefly Algorithm diimplementasikan sebagai metode optimasi untuk menentukan konfigurasi *pipeline scheduler* pada proses scraping dan *image text swapping*. Algoritma ini digunakan untuk mengoptimalkan jumlah

proses yang dapat berjalan secara bersamaan (*concurrency limit*) pada setiap tahapan pemrosesan data sehingga penggunaan sumber daya sistem menjadi lebih efisien.

Setiap individu (*firefly*) merepresentasikan satu konfigurasi *concurrency limit* yang terdiri atas lima variabel keputusan (*decision variables*), yaitu batas jumlah proses ekstraksi URL gambar dari *Document Object Model* (DOM), pengunduhan gambar, *Optical Character Recognition* (OCR), penerjemahan menggunakan Google Translate API, dan *image text swapping*. Setiap konfigurasi memiliki kombinasi nilai yang berbeda dan akan dievaluasi untuk memperoleh konfigurasi yang memberikan performa terbaik bagi sistem.

Kualitas setiap konfigurasi diukur menggunakan fungsi *fitness* yang mempertimbangkan beberapa indikator performa *pipeline*, yaitu *throughput*, *makespan*, *pipeline balance*, dan *retry count*. Konfigurasi dengan nilai *fitness* terbaik akan dipilih sebagai konfigurasi *scheduler* yang digunakan dalam proses pemrosesan data.

Proses optimasi dilakukan secara adaptif melalui mekanisme *Performance Monitor* yang memantau kondisi *pipeline* selama sistem berjalan. Ketika terdeteksi perubahan performa, seperti penurunan *throughput* atau ketidakseimbangan beban kerja antar tahapan, Firefly Algorithm akan dijalankan kembali untuk memperoleh konfigurasi *scheduler* yang lebih sesuai. Konfigurasi terbaik kemudian diterapkan secara dinamis tanpa menghentikan proses yang sedang berlangsung sehingga sistem mampu menyesuaikan diri terhadap perubahan beban kerja.

Melalui penerapan tersebut, Firefly Algorithm diharapkan mampu meningkatkan efisiensi proses scraping dan *image text swapping* dengan mengoptimalkan pemanfaatan sumber daya pada setiap tahapan *pipeline*. Selain mempercepat proses pemrosesan data, mekanisme ini juga membantu menjaga keseimbangan beban kerja antar proses sehingga kinerja sistem menjadi lebih stabil.

Dalam penelitian ini, Firefly Algorithm tidak digunakan untuk proses pengolahan citra maupun peningkatan akurasi OCR, melainkan diterapkan sebagai algoritma optimasi metaheuristik pada komponen *Pipeline Scheduler*. Algoritma ini bertugas mengoptimalkan urutan eksekusi tugas (*task scheduling*) dan alokasi daya pada *backend server* agar waktu pemrosesan total (*makespan*) dari rangkaian proses scraping, OCR, translasi, hingga *image text swapping* dapat berjalan secara efisien.

2.6 Ekstensi Chrome (Manifest V3)

Ekstensi Chrome (*Chrome Extension*) merupakan perangkat lunak berbasis peramban (*browser*) yang dikembangkan untuk menambahkan fungsi tertentu pada

Google Chrome. Ekstensi ini memungkinkan pengembang memperluas kemampuan *browser* melalui integrasi dengan halaman web, antarmuka pengguna, maupun berbagai *Application Programming Interface* (API) yang disediakan oleh Chrome. Menurut Google Chrome for Developers (2024), arsitektur terbaru Manifest V3 (MV3) dikembangkan dengan fokus pada peningkatan keamanan, privasi, serta efisiensi penggunaan sumber daya dibandingkan Manifest V2.

Pemilihan Ekstensi Chrome berbasis Manifest V3 sebagai komponen *client-side* dalam penelitian ini didasari oleh beberapa alasan teknis utama:

1. Akses Langsung ke *Rendered DOM* secara *Real-Time*

Situs *e-commerce* 1688.com menerapkan proteksi pemuatan dinamis berbasis JavaScript. Dengan berjalan sebagai ekstensi peramban, *Content Script* memiliki akses *native* langsung ke *Document Object Model* (DOM) yang telah selesai dirender oleh *browser* pengguna, sehingga mengeliminasi kendala kegagalan ekstraksi data akibat *dynamic rendering* tanpa perlu memicu sistem perlindungan *anti-bot/anti-scraping* dari server target (Chandra & Tiwari, 2021; Varghese & Vijay, 2019).

2. Efisiensi Penggunaan Sumber Daya Sistem

Berbeda dengan teknik *scraping* tradisional berbasis *server-side headless browser* (seperti Selenium atau Puppeteer) yang membutuhkan konsumsi memori (RAM) dan CPU sangat tinggi untuk menjalankan *instance browser* di *server*, penggunaan Ekstensi Chrome memanfaatkan *browser* yang sedang digunakan oleh pengguna (*client-side execution*). Hal ini secara signifikan mengurangi beban komputasi dan *overhead* pada *server backend* (Klosowski et al., 2022).

3. Integrasi Antarmuka yang Interaktif dan *User-Friendly*

Ekstensi Chrome memungkinkan integrasi tombol dan panel kontrol secara langsung (*DOM injection*) pada halaman produk 1688.com, sehingga pengguna dapat memicu proses *scraping*, OCR, hingga penerjemahan secara presisi hanya dengan satu klik tanpa harus berpindah aplikasi (Lam, 2021).

Manifest V3 memperkenalkan perubahan mendasar pada arsitektur ekstensi Chrome dengan menggantikan mekanisme *background page* menjadi *Service Worker* yang bersifat *event-driven*. Pendekatan tersebut memungkinkan ekstensi hanya dijalankan ketika diperlukan sehingga mampu mengurangi penggunaan memori dan meningkatkan efisiensi pemrosesan (Google Chrome for Developers, 2024). Selain itu, Manifest V3 juga memperketat mekanisme perizinan

(*permissions*) serta membatasi eksekusi kode dinamis sebagai upaya meningkatkan keamanan ekstensi.

Secara umum, arsitektur Chrome Extension pada Manifest V3 terdiri atas beberapa komponen utama, yaitu Content Script, Service Worker, dan Extension User Interface. Content Script merupakan skrip JavaScript yang dijalankan pada halaman web sehingga memiliki akses terhadap Document Object Model (DOM) untuk membaca maupun memodifikasi elemen halaman sesuai izin yang diberikan. Sementara itu, Service Worker bertugas menangani proses di latar belakang (*background processing*), seperti komunikasi dengan layanan eksternal, pengelolaan penyimpanan data, serta pengendalian logika aplikasi. Adapun Extension User Interface menyediakan antarmuka interaksi antara pengguna dengan ekstensi melalui *popup*, ikon toolbar, maupun halaman pengaturan (*options page*) (Lam, 2021).

Komunikasi antar komponen pada Manifest V3 dilakukan menggunakan mekanisme Message Passing, yaitu pertukaran pesan antara *Content Script*, *Service Worker*, maupun komponen antarmuka pengguna. Mekanisme ini memungkinkan setiap komponen menjalankan tugas sesuai fungsinya tanpa saling bergantung secara langsung sehingga meningkatkan modularitas dan efisiensi pengembangan sistem (Google Chrome for Developers, 2024).

Manifest V3 merupakan standar terbaru dalam pengembangan Chrome Extension yang menggantikan Manifest V2 dengan peningkatan pada aspek keamanan, privasi, serta efisiensi pengelolaan proses latar belakang (*background processing*). Selain itu, Manifest V3 tetap menyediakan mekanisme Content Script, Service Worker, dan Message Passing yang menjadi komponen utama dalam pengembangan ekstensi modern (Google Chrome for Developers, 2024).

2.7 Pipeline Scheduler

Pipeline Scheduler merupakan komponen pengatur alur kerja (*workflow execution manager*) yang bertugas mengelola, menjadwalkan, serta mengoordinasikan urutan eksekusi tugas-tugas (*tasks*) dalam sebuah sistem pemrosesan data bertahap (*multi-stage processing pipeline*) (Deelman et al., 2019). Dalam arsitektur pemrosesan data terintegrasi seperti rantai proses yang melibatkan ekstraksi DOM, pengunduhan citra, Optical Character Recognition (OCR), ekstraksi terjemahan, dan rekonstruksi citra (*image text swapping*) *scheduler* mengontrol alokasi sumber daya, tingkat eksekusi sejajar (*concurrency level*), antrean tugas (*task queuing*), serta penanganan kueri berulang saat terjadi kegagalan (*retry mechanism*) (Silberschatz et al., 2018).

Penjadwalan *pipeline* yang efisien dirancang untuk meminimalkan waktu penyelesaian total (*makespan*) dan meningkatkan jumlah tugas yang terselesaikan per satuan waktu (*throughput*). Dalam lingkungan pemrosesan asinkron yang melibatkan panggilan API pihak ketiga (seperti Google Translate API dan LLM), *pipeline scheduler* mencegah terjadinya kemacetan pemrosesan (*bottlenecking*) dan menjaga stabilitas sistem agar tidak melampaui batas laju permintaan (*rate limit*) maupun menguras memori (*resource exhaustion*) (Klosowski et al., 2022).

2.8 Flask

Flask merupakan sebuah *micro-framework* berbasis bahasa pemrograman Python yang dikembangkan oleh Armin Ronacher di bawah bendera Pocco (Grinberg, 2018). Didesain dengan prinsip ketersediaan komponen inti yang minimalis (*lightweight*), Flask tidak mewajibkan pustaka atau alat (*tools*) tertentu dan tidak menyediakan lapisan abstraksi database secara bawaan. Hal ini memberikan fleksibilitas tinggi bagi pengembang untuk memilih dan mengintegrasikan modul ekstensi sesuai kebutuhan spesifik aplikasi (Kroch, 2021).

Flask mengandalkan dua pustaka utama, yaitu WSGI (*Web Server Gateway Interface*) toolkit bernama Werkzeug untuk menangani aspek komunikasi jaringan dan routing HTTP, serta Jinja2 sebagai mesin pembuat templat (*template engine*) (Ronacher, 2020). Karena keunggulannya yang memiliki jejak memori (*overhead*) minim, Flask sangat cocok digunakan untuk membangun layanan antarmuka pemrograman aplikasi (*Application Programming Interface* / API) berbasis RESTful, khususnya pada sistem berbasis Python (Grinberg, 2018).

Dalam arsitektur penelitian ini, Flask difungsikan sebagai penyedia *backend server* yang menerima permintaan (*request*) dari ekstensi peramban (*client*). Flask bertindak sebagai pengatur alur kerja yang menjembatani pertukaran data JSON dan mengoordinasikan eksekusi modul-modul pemrosesan berat di sisi *server*, seperti pemrosesan OCR, integrasi API penerjemah, algoritma *Image Text Swapping*, komunikasi dengan LLM Gemini, serta modul optimasi *Pipeline Scheduler* berbasis *Firefly Algorithm* (Kroch, 2021).

2.9 Platform 1688

1688.com merupakan platform perdagangan elektronik (*e-commerce*) berbasis Business-to-Business (B2B) yang dimiliki oleh Alibaba Group. Platform ini menghubungkan produsen dan pemasok di Tiongkok dengan pelaku usaha, distributor, maupun importir yang membutuhkan produk dalam jumlah besar. Dengan banyaknya variasi produk serta harga yang kompetitif, 1688.com menjadi salah satu platform yang banyak dimanfaatkan sebagai sumber pengadaan produk oleh pelaku usaha impor dari berbagai negara.

13 Sebagai ilustrasi, struktur halaman produk pada platform 1688.com ditunjukkan pada Gambar 2.3.



Gambar 2.3 Struktur Web 1688

Sebagian besar antarmuka, informasi produk, serta komunikasi pada platform 1688.com disajikan dalam bahasa Mandarin sehingga menjadi salah satu kendala bagi pengguna internasional yang tidak memahami bahasa tersebut (Fasdana, 2025). Selain hambatan bahasa, informasi produk pada 1688.com juga memiliki karakteristik yang berbeda dibandingkan platform *e-commerce* lainnya. Sebagian informasi penting, seperti spesifikasi produk, ukuran, maupun informasi promosi, sering kali disajikan dalam bentuk teks yang menyatu pada gambar produk (*in-image text*). Di sisi lain, informasi produk pada halaman web dimuat secara dinamis menggunakan JavaScript sehingga proses pengambilan data memerlukan pendekatan yang mampu membaca struktur halaman setelah proses *rendering* selesai.

Karakteristik tersebut menjadikan 1688.com sebagai platform yang memiliki tantangan tersendiri dalam proses pengambilan data maupun pemahaman informasi produk. Oleh karena itu, berbagai pendekatan seperti *web scraping*, *Optical Character Recognition (OCR)*, dan penerjemahan otomatis banyak dimanfaatkan untuk membantu pengguna memperoleh informasi produk secara lebih mudah. Pada penelitian ini, karakteristik tersebut menjadi dasar dalam pengembangan sistem untuk mendukung penyusunan katalog produk multibahasa.

88 2.10 Unified Modeling Language (UML)

Menurut Booch, Rumbaugh, dan Jacobson (2023), Unified Modeling Language (UML) adalah bahasa pemodelan standar yang digunakan untuk merancang, memvisualisasikan, dan mendokumentasikan sistem perangkat lunak berbasis *object-oriented*. UML berfungsi sebagai alat bantu untuk memahami

struktur sistem secara konseptual sebelum tahap implementasi dilakukan. Dengan UML, pengembang dapat mengidentifikasi interaksi antar komponen, alur data, serta hubungan antar proses yang terjadi di dalam sistem.

UML banyak digunakan dalam pengembangan perangkat lunak modern karena mampu memberikan gambaran sistem yang lebih terstruktur dan mudah dipahami.

Seperti dijelaskan oleh López & García (2021), UML membantu dalam mengurangi ambiguitas pada tahap analisis kebutuhan dan meningkatkan kolaborasi antar pengembang, analis, dan pengguna akhir.

Dalam penelitian ini, UML digunakan untuk menggambarkan interaksi antara pengguna (importir) dan sistem yang dikembangkan, yaitu sistem ekstensi Chrome Firefly Algorithm untuk optimasi proses scraping dan image text swapping pada penyusunan katalog produk multibahasa platform 1688.

2.10.1 Activity Diagram

Activity Diagram menggambarkan aliran kerja (*workflow*) atau urutan aktivitas dalam sistem, mulai dari titik awal (*initial state*), mengeksekusi serangkaian alur keputusan (*decision nodes*), hingga mencapai kondisi akhir (*final state*) (Dumas et al., 2018). Dalam penelitian ini, *activity diagram* difungsikan untuk memetakan alur eksekusi data secara mendalam, seperti alur pembacaan DOM oleh *Content Script* Ekstensi Chrome, pengiriman *payload* HTTP ke *Flask REST API*, mekanisme antrean tugas pada *Pipeline Scheduler*, hingga rekonstruksi citra produk.

2.10.2 Class Diagram

Class Diagram memodelkan struktur statis sistem dengan menampilkan kelas-kelas (*classes*), atribut, metode/fungsi (*methods*), serta hubungan asosiasi maupun himpunan (*inheritance/aggregation*) antarkelas dalam perangkat lunak (Pressman & Maxim, 2020). Diagram ini digunakan untuk merancang struktur data *backend* Python (Flask), modul pengelola *EasyOCR*, struktur entitas data produk 1688, serta variabel representasi posisi dan atribut dalam *Firefly Algorithm*.

2.10.3 Sequence Diagram

Sequence Diagram memodelkan interaksi antar objek berdasarkan urutan waktu (*time-ordered interaction*) (Sommerville, 2016). Diagram ini memberikan visualisasi yang jelas mengenai bagaimana komponen-komponen terpisah seperti *Browser UI/Content Script*, *Service Worker (Background)*, *Flask REST API Server*, *Pipeline Scheduler*, dan API eksternal (Google Translate/Gemini) saling bertukar

pesan (*messages/HTTP requests*) secara asinkron dalam satu siklus pemrosesan lengkap.

2.11 Large Language Model (LLM) Gemini

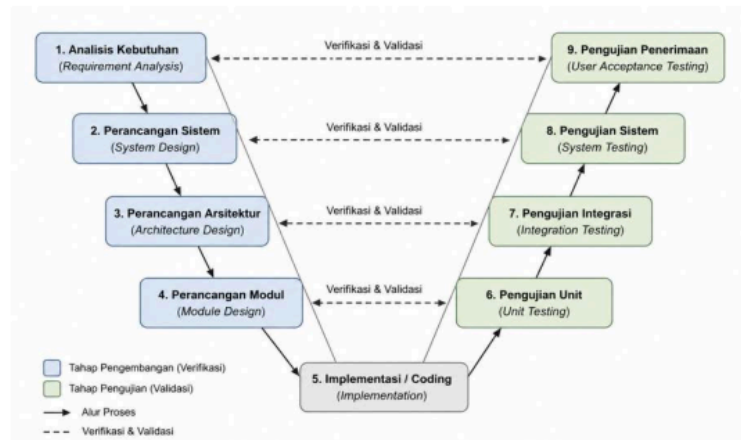
⁵ Gemini merupakan model bahasa besar (*Large Language Model* / LLM) yang dikembangkan oleh Google DeepMind. Model ini dirancang secara multimodal sehingga mampu memahami berbagai jenis masukan, seperti ⁵ teks, gambar, audio, video, dan kode dalam satu arsitektur terpadu (Google DeepMind, 2023).

Salah satu kemampuan utama Gemini adalah menghasilkan teks yang koheren, memahami konteks semantik, melakukan penalaran terhadap informasi yang diberikan, serta menyusun ringkasan maupun deskripsi berdasarkan instruksi pengguna. Kemampuan tersebut menjadikan Gemini banyak dimanfaatkan pada berbagai aplikasi berbasis kecerdasan buatan generatif yang memerlukan penyusunan informasi secara otomatis.

Dalam penelitian ini, Gemini dimanfaatkan untuk menghasilkan ringkasan deskripsi produk berbahasa Indonesia berdasarkan informasi hasil ekstraksi OCR dan metadata produk yang diperoleh melalui proses scraping. Dengan pendekatan tersebut, sistem mampu menyajikan deskripsi produk yang lebih ringkas, mudah dipahami, dan sesuai dengan konteks produk sehingga dapat membantu pengguna dalam melakukan analisis sebelum proses pembelian.

2.12 ⁷⁷ V-Model (Verification and Validation Model)

¹⁰² *Verification and Validation Model* (V-Model) merupakan salah satu model *Software Development Life Cycle (SDLC)* yang menekankan hubungan antara setiap tahapan pengembangan perangkat lunak dengan tahapan pengujian yang sesuai. Berbeda dengan model Waterfall yang memisahkan proses pengembangan dan pengujian, V-Model mengintegrasikan aktivitas verifikasi dan validasi sejak awal proses ²⁹ pengembangan sehingga setiap tahapan dapat diuji secara sistematis. Pendekatan ini bertujuan untuk memastikan bahwa perangkat lunak dibangun sesuai dengan kebutuhan pengguna serta memenuhi spesifikasi yang telah ditetapkan. (OpenStax, 2024).



Gambar 2.4 Verification and Validation Model (V-Model)

Sumber: Diadaptasi dari OpenStax (2024).

Gambar 2.4 menunjukkan tahapan V-Model yang terdiri atas dua sisi utama. Sisi kiri menggambarkan tahapan pengembangan perangkat lunak yang dimulai dari analisis kebutuhan, perancangan sistem, perancangan arsitektur, hingga perancangan modul. Setelah seluruh rancangan selesai, proses dilanjutkan pada tahap implementasi (*implementation/coding*) sebagai titik penghubung kedua sisi model.

Setelah proses implementasi selesai, tahapan pada sisi kanan dilakukan sebagai proses validasi terhadap setiap hasil pengembangan. Pengujian dimulai dari *unit testing*, dilanjutkan dengan *integration testing*, *system testing*, hingga *user acceptance testing*. Setiap tahapan pengujian memiliki keterkaitan langsung dengan tahapan perancangan pada sisi kiri sehingga kesalahan yang ditemukan dapat ditelusuri kembali ke tahap pengembangan yang bersesuaian.

Pada penelitian ini, V-Model dipilih sebagai metode pengembangan perangkat lunak karena mampu menghubungkan setiap tahapan perancangan dengan proses pengujian yang sesuai. Pendekatan ini memudahkan proses verifikasi terhadap setiap fitur yang dikembangkan, seperti proses scraping halaman produk, ekstraksi teks menggunakan OCR, penerjemahan menggunakan Google Translate API, image text swapping, penyusunan deskripsi produk menggunakan Gemini, serta optimasi pipeline menggunakan Firefly Algorithm. Dengan demikian, setiap komponen sistem dapat diuji secara bertahap sehingga mengurangi kemungkinan terjadinya kesalahan pada tahap implementasi akhir.

2.13 API Google Translate

Google Translate API merupakan layanan penerjemahan mesin (*Machine Translation*) yang disediakan oleh Google Cloud. API ini memungkinkan aplikasi

menerjemahkan teks secara otomatis ke berbagai bahasa melalui antarmuka berbasis REST tanpa memerlukan pembangunan model penerjemahan secara mandiri. Google Translate API mendukung lebih dari 100 bahasa dan memanfaatkan teknologi *Neural Machine Translation (NMT)* untuk menghasilkan terjemahan yang lebih alami dan mempertahankan konteks kalimat.

Dalam penelitian ini, Google Translate API digunakan untuk menerjemahkan teks berbahasa Mandarin yang diperoleh dari proses OCR menjadi bahasa Indonesia. Hasil terjemahan tersebut selanjutnya digunakan pada proses *image text swapping*, yaitu menggantikan teks Mandarin pada gambar produk dengan teks hasil terjemahan sehingga informasi produk dapat dipahami secara langsung oleh pengguna ketika mengakses halaman 1688.com.

2.14 Penelitian Terdahulu

Sebelum memulai penelitian ini, penulis mencari referensi untuk melakukan penelitian dengan studi literatur mengenai penelitian terkait yang telah dilakukan oleh peneliti sebelumnya. Berikut adalah tabel yang berisikan beberapa penelitian terdahulu yang digunakan untuk penulis menjadi referensi dalam melakukan penelitian.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti dan Tahun	Fokus Penelitian	Temuan Utama	Keterkaitan dengan Penelitian Ini
1	Chandra & Tiwari (2021)	Web Scraping berbasis DOM pada aplikasi web dinamis berbasis JavaScript.	Ekstraksi data menggunakan DOM Scraping terbukti lebih adaptif dan presisi pada halaman web dinamis berbasis JavaScript dibandingkan teknik <i>static HTML fetching</i> .	Menjadi dasar penggunaan teknik DOM Scraping via Content Script untuk mengekstraksi data produk pada platform 1688.com.
2	Chen et al. (2022)	Pipeline pengolahan citra (OCR dan Image Text Swapping) untuk lokalisasi teks.	Penggabungan deep learning OCR dan teknik inpainting/text overlay mampu mengganti teks visual pada citra tanpa merusak tekstur latar belakang (background).	Menjadi landasan penyusunan pipeline pemrosesan visual (EasyOCR dan Image Text Swapping) untuk mentransformasi teks Mandarin pada gambar produk 1688.

3	Klosowski et al. (2022)	Optimasi penjadwalan <i>task/scheduler</i> pada arsitektur pemrosesan asinkron menggunakan <i>Firefly Algorithm</i> .	<i>Firefly Algorithm</i> efektif meminimalkan waktu penyelesaian total (<i>makespan</i>) dan mencegah <i>bottleneck</i> dalam penjadwalan antrean <i>task</i> paralel.	Menjadi dasar penerapan <i>Firefly Algorithm</i> untuk mengoptimalkan urutan eksekusi tugas (<i>Pipeline Scheduler</i>) di <i>backend server</i> .
4	Lam (2021)	Pengembangan <i>Chrome Extension</i> menggunakan arsitektur Manifest V3 untuk ekstraksi data <i>real-time</i> .	Arsitektur Manifest V3 melalui <i>Content Script</i> dan <i>Service Worker</i> memberikan akses <i>native</i> ke <i>DOM client</i> secara efisien tanpa konsumsi memori <i>server</i> yang besar.	Menjadi rujukan dalam membangun antarmuka dan mekanisme <i>client-side processing</i> berbasis <i>Chrome Extension Manifest V3</i> .
5	Zhao & Wang (2023)	Sistem otomatisasi pemrosesan produk <i>e-commerce</i> mencakup <i>scraping</i> , <i>penerjemahan</i> , dan <i>OCR</i> .	Integrasi <i>scraping</i> , <i>penerjemahan</i> , dan <i>OCR</i> mampu mempercepat lokalisasi katalog produk, namun performa sistem menurun saat menangani antrean pemrosesan visual berukuran besar.	Menjadi acuan sistem yang paling mirip. Penelitian ini menyempurnakan keterbatasan tersebut dengan menambahkan <i>Image Text Swapping</i> dan optimasi <i>Pipeline Scheduler</i> berbasis <i>Firefly Algorithm</i> .

Berdasarkan kajian penelitian terdahulu pada Tabel 2.1, belum ditemukan penelitian yang mengintegrasikan *Chrome Extension* berbasis *DOM Scraping* dengan *pipeline OCR*, *Google Translate*, *Image Text Swapping*, serta optimasi

Pipeline Scheduler menggunakan *Firefly Algorithm* secara terpadu pada platform 1688.com.

133
Sebagian besar penelitian terdahulu hanya berfokus pada satu atau dua aspek secara terpisah seperti *scraping* halaman dinamis saja (Chandra & Tiwari, 2021), *image text swapping* secara independen (Chen et al., 2022), atau optimasi penjadwalan tugas tanpa penerapan pada *pipeline* citra multibahasa (Klosowski et al., 2022). Adapun penelitian yang paling mendekati (Zhao & Wang, 2023) telah mengintegrasikan *scraping*, penerjemahan, dan OCR, namun belum menerapkan teknik rekonstruksi citra (*image text swapping*) serta belum memiliki mekanisme optimasi penjadwalan *task* otomatis saat terjadi beban pemrosesan tinggi.

Oleh karena itu, kebaruan (*novelty*) dari penelitian ini terletak pada pengintegrasian seluruh komponen tersebut ke dalam satu ekosistem yang utuh. Penelitian ini memanfaatkan *Chrome Extension (Manifest V3)* sebagai *client* berbasis *DOM Scraping*, yang terhubung dengan *backend Flask* untuk menjalankan *pipeline* OCR, penerjemahan, dan *Image Text Swapping*, dengan alokasi urutan eksekusinya dioptimalkan secara dinamis oleh *Firefly Algorithm* pada *Pipeline Scheduler*. Integrasi ini dirancang untuk menghasilkan sistem penyusunan katalog produk multibahasa dari platform 1688.com yang efisien dan adaptif.

BAB 3. METODOLOGI PENELITIAN

3.1 Pendekatan Metodologi

Penelitian ini menggunakan pendekatan Research and Development (R&D) karena bertujuan untuk mengembangkan sebuah sistem yang dapat membantu pelaku usaha dalam memperoleh dan mengolah informasi produk dari platform 1688.com. Pendekatan ini dipilih karena tidak hanya berfokus pada analisis permasalahan, tetapi juga menghasilkan sebuah produk berupa sistem yang dapat digunakan untuk menyelesaikan permasalahan tersebut.

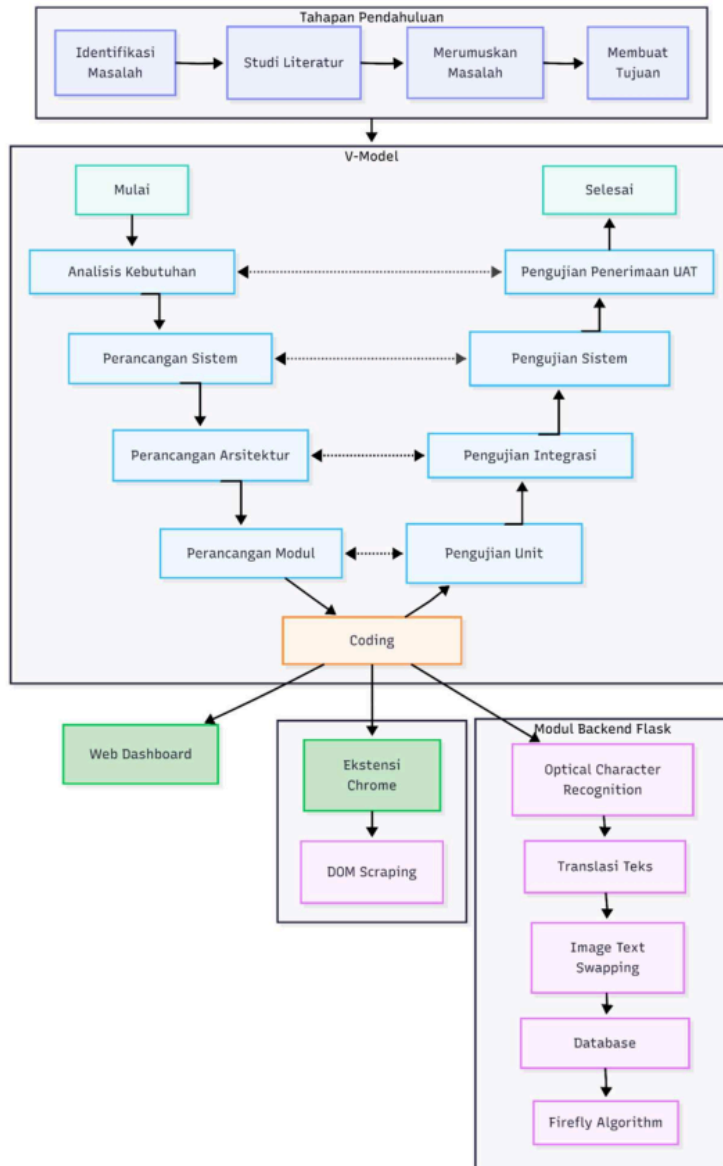
Model pengembangan sistem yang digunakan dalam penelitian ini adalah V-Model. Model ini dipilih karena memiliki tahapan pengembangan yang terstruktur, dimulai dari identifikasi kebutuhan, perancangan sistem, implementasi, hingga pengujian. Selain itu, setiap tahapan pengembangan memiliki proses verifikasi dan validasi sehingga sistem yang dikembangkan dapat dievaluasi secara sistematis untuk memastikan kesesuaiannya dengan kebutuhan pengguna.

3.2 Tahapan Penelitian

Penelitian ini menggunakan model pengembangan sistem V-Model sebagai acuan dalam proses pengembangan sistem. V-Model dipilih karena menyediakan tahapan pengembangan yang sistematis serta menghubungkan setiap proses perancangan dengan tahapan pengujian yang sesuai. Melalui pendekatan ini, setiap hasil yang diperoleh pada tahap pengembangan dapat diverifikasi dan divalidasi secara bertahap sehingga sistem yang dihasilkan sesuai dengan kebutuhan pengguna.

Sebelum memasuki proses pengembangan menggunakan V-Model, penelitian diawali dengan tahap pendahuluan yang terdiri atas identifikasi masalah, studi literatur, perumusan masalah, dan penetapan tujuan penelitian. Tahap ini bertujuan untuk memahami permasalahan penelitian, menentukan ruang lingkup penelitian, serta memperoleh landasan teori dan metode yang relevan sebagai dasar dalam pengembangan sistem.

Selanjutnya proses pengembangan sistem dilakukan menggunakan metode V-Model yang terdiri atas tahapan analisis kebutuhan sistem, perancangan sistem, perancangan arsitektur, perancangan modul, implementasi (*coding*), serta pengujian yang meliputi Unit Testing, Integration Testing, System Testing, dan User Acceptance Testing (UAT). Hubungan antara tahapan penelitian dan proses pengembangan sistem menggunakan V-Model ditunjukkan pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian Menggunakan V-Model

Berdasarkan Gambar 3.1, penelitian ini terdiri atas dua bagian utama, yaitu tahap pendahuluan dan tahap pengembangan sistem menggunakan V-Model. Tahap pendahuluan menghasilkan dasar penelitian berupa identifikasi permasalahan, rumusan masalah, tujuan penelitian, serta kajian literatur yang selanjutnya digunakan sebagai acuan pada proses pengembangan sistem.

Tahap pertama pada V-Model adalah analisis kebutuhan sistem, yaitu mengidentifikasi kebutuhan fungsional dan nonfungsional yang harus dipenuhi oleh sistem berdasarkan hasil identifikasi masalah dan studi literatur. Hasil dari tahap ini menjadi dasar dalam penyusunan rancangan sistem.

Tahap berikutnya adalah perancangan sistem, yaitu menyusun rancangan solusi berdasarkan kebutuhan yang telah diperoleh. Pada tahap ini dilakukan penyusunan gambaran umum sistem, proses bisnis, serta kebutuhan fungsional yang akan diimplementasikan.

Selanjutnya dilakukan perancangan arsitektur (*Architecture Design*) untuk menentukan struktur sistem beserta hubungan antar komponen utama. Hasil dari tahap ini berupa rancangan arsitektur yang terdiri atas Chrome Extension, Backend Flask, dan Web Dashboard sebagai komponen utama sistem.

Setelah rancangan arsitektur selesai disusun, dilakukan perancangan modul (*Module Design*) dengan mendefinisikan fungsi setiap modul yang akan dikembangkan. Modul yang dirancang meliputi DOM Scraping pada Chrome Extension, *Optical Character Recognition (OCR)*, Google Translate API, *Image Text Swapping*, *Database*, serta *Firefly Algorithm* pada *Backend Flask*.

Tahap implementasi (*Coding*) merupakan proses pembangunan sistem berdasarkan seluruh hasil perancangan yang telah disusun. Implementasi menghasilkan tiga komponen utama, yaitu Web Dashboard sebagai media pengolahan data hasil scraping, Chrome Extension sebagai antarmuka pengguna pada platform 1688.com yang menjalankan proses DOM Scraping, serta Modul Backend Flask yang mengimplementasikan proses Optical Character Recognition (OCR), penerjemahan teks, Image Text Swapping, pengelolaan database, dan Firefly Algorithm sebagai mekanisme optimasi pada sistem.

Setelah proses implementasi selesai, sistem memasuki tahap pengujian dan validasi yang terdiri atas *Unit Testing*, *Integration Testing*, *System Testing*, dan *User Acceptance Testing (UAT)*. Pengujian dilakukan secara bertahap untuk memastikan bahwa setiap modul, integrasi antar komponen, serta keseluruhan fungsi sistem telah berjalan sesuai dengan kebutuhan pengguna dan tujuan penelitian.

Dengan demikian, setiap tahapan pengembangan pada sisi kiri V-Model memiliki keterkaitan langsung dengan tahapan pengujian pada sisi kanan. Pendekatan ini memastikan bahwa setiap hasil perancangan telah melalui proses verifikasi dan validasi sebelum sistem digunakan. Hubungan antara tahapan pengembangan dan tahapan pengujian pada V-Model dirangkum pada Tabel 3.1.

Tabel 3.1 Hubungan Tahapan Pengembangan dan Tahapan Pengujian pada V-Model

Fase Perancangan (Kiri)	Hubungan Pengujian (Kanan)	Penjelasan Hubungan
Analisis Kebutuhan Sistem	Pengujian Penerimaan (UAT)	Pengujian dilakukan berdasarkan kebutuhan pengguna untuk memastikan sistem memenuhi ekspektasi dan tujuan awal.
Perancangan Sistem	Pengujian Sistem	Menguji apakah sistem secara keseluruhan bekerja sesuai rancangan fungsional dan non-fungsional yang telah dibuat.
Arsitektur Design	Pengujian Integrasi	Memastikan modul-modul yang dirancang dapat berkomunikasi dan terintegrasi tanpa error.
Module Design	Pengujian Unit	Menguji setiap modul atau komponen kecil (seperti scraping, OCR, translasi) agar berfungsi sesuai spesifikasi sebelum digabungkan.
Coding (Implementasi)	—	Tahap implementasi utama yang menjadi jembatan antara perancangan dan pengujian seluruh sistem.

² 3.2.1 Identifikasi Masalah dan Studi Literatur

Tahap pertama dalam penelitian ini adalah melakukan identifikasi masalah dan studi literatur. Identifikasi masalah bertujuan untuk memperoleh pemahaman mengenai kondisi yang dihadapi pelaku usaha dalam memperoleh informasi produk dari platform 1688.com serta mengidentifikasi kendala yang muncul selama proses tersebut.

Identifikasi masalah dilakukan melalui observasi terhadap platform 1688.com dan proses bisnis pengadaan produk yang dilakukan oleh pelaku usaha. Hasil observasi menunjukkan bahwa sebagian besar informasi penting mengenai spesifikasi, keunggulan, serta deskripsi produk disajikan dalam bentuk gambar yang mengandung teks berbahasa Mandarin. Kondisi tersebut menyebabkan informasi tidak dapat diterjemahkan secara langsung menggunakan penerjemah

berbasis halaman web, sehingga proses analisis produk ⁵³menjadi kurang efektif dan memerlukan waktu yang lebih lama.

Selanjutnya dilakukan ⁴²studi literatur dengan mempelajari berbagai referensi ilmiah yang berkaitan dengan *web scraping*, *Optical Character Recognition* (OCR), Google Translate API, *image text swapping*, Large Language ⁶⁰Model (LLM), Chrome Extension, serta Firefly Algorithm. Studi literatur ⁶⁰dilakukan untuk memperoleh landasan teoritis, memahami perkembangan penelitian terdahulu, serta menentukan metode dan teknologi yang sesuai dengan kebutuhan penelitian.

¹⁰Hasil dari tahap ini berupa rumusan permasalahan, tujuan penelitian, serta dasar pemilihan metode dan teknologi yang digunakan dalam pengembangan sistem.

3.2.2 Analisis Kebutuhan Sistem

Setelah permasalahan penelitian berhasil diidentifikasi, tahap selanjutnya adalah melakukan ²⁷analisis kebutuhan sistem. Tahap ini bertujuan untuk mengidentifikasi kebutuhan yang harus dipenuhi oleh sistem agar mampu menyelesaikan permasalahan yang ditemukan selama proses observasi.

Analisis kebutuhan dilakukan berdasarkan hasil observasi terhadap platform 1688.com, studi literatur, serta identifikasi proses bisnis yang dilakukan oleh pelaku usaha dalam memperoleh informasi produk impor. ¹⁵Pada tahap ini dilakukan identifikasi terhadap kebutuhan fungsional dan nonfungsional sistem, termasuk kebutuhan terkait proses pengambilan data produk (*web scraping*), ekstraksi teks pada gambar menggunakan *Optical Character Recognition* (OCR), penerjemahan informasi produk, pengelolaan data hasil scraping, serta penyajian informasi melalui dashboard.

³3.2.3 Perancangan Sistem

Tahap perancangan sistem dilakukan setelah kebutuhan sistem berhasil diidentifikasi. Tujuan dari ³tahap ini adalah menyusun rancangan sebagai acuan dalam proses pengembangan sistem sehingga implementasi dapat dilakukan secara terstruktur dan sesuai dengan kebutuhan yang telah ditetapkan.

⁸⁶Pada tahap ini dilakukan perancangan terhadap arsitektur sistem, alur proses, basis data, antarmuka pengguna, serta mekanisme integrasi antar komponen sistem. Perancangan juga mencakup penyusunan alur proses *web scraping*, ekstraksi teks pada gambar menggunakan *Optical Character Recognition* (OCR), proses penerjemahan informasi produk, serta pengelolaan data hasil scraping yang akan disajikan melalui dashboard.

3.2.4 Implementasi Sistem

Tahap implementasi sistem merupakan proses penerapan hasil perancangan ke dalam bentuk aplikasi yang dapat dijalankan. Pada tahap ini seluruh rancangan yang telah disusun pada tahap sebelumnya direalisasikan menjadi sebuah sistem yang berfungsi sesuai dengan kebutuhan yang telah ditetapkan.

Implementasi dilakukan dengan mengembangkan setiap komponen sistem secara bertahap, meliputi ekstensi Google Chrome sebagai media interaksi pengguna, aplikasi web sebagai dashboard pengelolaan data, serta layanan backend yang bertugas mengelola proses pengambilan data produk, pengolahan informasi, dan penyimpanan data. Seluruh komponen tersebut diintegrasikan sehingga mampu bekerja sebagai satu kesatuan sistem.

3.2.5 Pengujian dan Validasi

Tahap terakhir dalam penelitian ini adalah pengujian dan validasi sistem. Tahap ini bertujuan untuk memastikan bahwa sistem yang telah diimplementasikan dapat berfungsi sesuai dengan kebutuhan yang telah ditetapkan pada tahap analisis serta memenuhi tujuan penelitian.

Pengujian dilakukan secara bertahap mengikuti pendekatan V-Model, yang meliputi *Unit Testing*, *Integration Testing*, *System Testing*, dan *User Acceptance Testing* (UAT). *Unit Testing* dilakukan untuk memastikan setiap modul sistem dapat berfungsi dengan baik secara mandiri. *Integration Testing* bertujuan untuk memverifikasi bahwa seluruh modul dapat saling terintegrasi dan berkomunikasi dengan baik. Selanjutnya, *System Testing* dilakukan untuk menguji keseluruhan fungsi sistem sesuai dengan kebutuhan yang telah dirancang. Tahap terakhir adalah *User Acceptance Testing* (UAT) yang dilakukan untuk mengevaluasi apakah sistem telah memenuhi kebutuhan pengguna dan dapat digunakan sesuai dengan tujuan pengembangannya. Melalui proses pengujian dan validasi tersebut, sistem diharapkan mampu bekerja secara optimal dalam mendukung proses pengambilan informasi produk dari platform 1688.com.

3.2.6 Teknik Pengumpulan Data

Teknik pengumpulan data merupakan tahapan yang dilakukan untuk memperoleh informasi yang dibutuhkan sebagai dasar dalam analisis kebutuhan sistem. Data yang dikumpulkan pada penelitian ini berasal dari berbagai sumber, baik berupa referensi ilmiah maupun hasil observasi langsung terhadap platform 1688.com. Teknik pengumpulan data yang digunakan meliputi studi literatur, observasi, dan dokumentasi.

¹3.2.7 Studi Literatur

Studi literatur dilakukan dengan mempelajari berbagai sumber ilmiah, seperti buku, jurnal, prosiding, dokumentasi resmi, dan penelitian terdahulu yang berkaitan dengan topik penelitian. Literatur yang dikaji mencakup teknologi *web scraping*, Optical Character Recognition (OCR), Google Translate API, image text swapping, Large Language Model (LLM), Chrome Extension, Firefly Algorithm, serta teknologi pendukung lainnya yang relevan dengan pengembangan sistem.

Hasil studi literatur digunakan sebagai landasan teoritis dalam penelitian, membantu pemilihan metode yang sesuai, serta menjadi acuan dalam proses pengembangan dan evaluasi sistem yang dibangun.

3.2.8 Observasi

Observasi dilakukan secara langsung terhadap platform 1688.com untuk memahami proses pencarian dan pengelolaan informasi produk oleh pelaku usaha impor. Kegiatan observasi meliputi pengamatan terhadap alur pencarian produk, struktur halaman produk, mekanisme pemuatan data pada halaman web, serta informasi produk yang tersedia dalam bentuk teks maupun gambar.

Selain mengamati karakteristik platform, observasi juga dilakukan terhadap proses penyusunan katalog produk, mulai dari pencarian produk, penerjemahan informasi, hingga penyusunan data katalog. Hasil observasi tersebut digunakan sebagai dasar dalam mengidentifikasi permasalahan pada sistem berjalan serta merumuskan kebutuhan sistem yang akan dikembangkan.

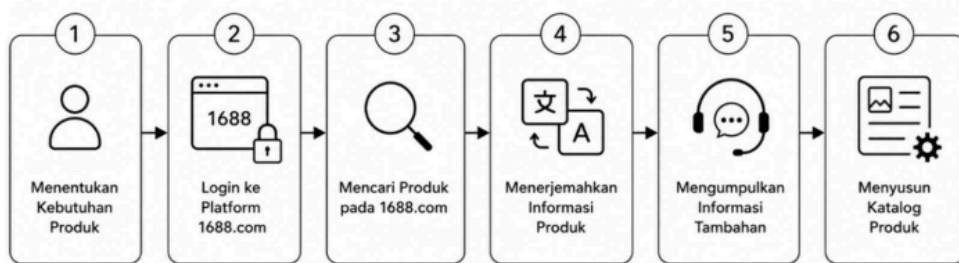
⁵²3.3 Analisis Sistem

Analisis sistem merupakan tahapan yang dilakukan untuk memahami proses bisnis yang berjalan, mengidentifikasi permasalahan, serta merumuskan kebutuhan sistem yang akan dikembangkan. Tahap ini dilakukan berdasarkan hasil observasi terhadap platform 1688.com serta proses pencarian, pengumpulan, dan pengelolaan informasi produk yang dilakukan oleh pelaku usaha impor. Selain itu, analisis juga mempertimbangkan kebutuhan pengguna terhadap proses *web scraping*, penerjemahan informasi produk, dan *image text swapping*. Hasil analisis tersebut menjadi dasar dalam penyusunan rancangan sistem yang dibahas pada Bab IV.

⁷3.3.1 Analisis Proses Bisnis

Analisis proses bisnis dilakukan untuk memahami alur kerja yang saat ini diterapkan oleh pelaku usaha dalam memperoleh informasi produk dari platform 1688.com hingga menyusun katalog produk. Analisis ini bertujuan untuk

mengidentifikasi aktivitas yang masih dilakukan secara manual sehingga dapat diketahui peluang penerapan sistem dalam meningkatkan efisiensi proses kerja.



Gambar 3.2 Alur Proses Bisnis Pengadaan Data Produk dari Platform 1688.com

Berdasarkan Gambar 3.2, proses bisnis diawali dengan penentuan kebutuhan produk oleh pelaku usaha, kemudian dilanjutkan dengan proses login ke platform 1688.com, pencarian produk, penerjemahan informasi produk, pengumpulan informasi tambahan dari supplier, hingga penyusunan katalog produk. Seluruh tahapan tersebut masih dilakukan secara manual dengan menggunakan beberapa aplikasi atau layanan yang berbeda sehingga proses pengumpulan dan pengolahan informasi produk memerlukan waktu yang relatif lama serta berpotensi menimbulkan kesalahan.

Berdasarkan analisis proses bisnis tersebut, diketahui bahwa proses pengolahan informasi produk masih melibatkan berbagai aktivitas manual, seperti pengambilan data produk, penerjemahan teks pada gambar, pengumpulan informasi tambahan, serta penyusunan katalog produk. Selain itu, penggunaan beberapa aplikasi atau layanan secara terpisah menyebabkan alur kerja menjadi kurang efisien serta meningkatkan potensi kesalahan dalam pengolahan data. Oleh karena itu, diperlukan suatu sistem yang mampu mengintegrasikan proses DOM Scraping, *Optical Character Recognition (OCR)*, penerjemahan teks, *Image Text Swapping*, serta pengelolaan data produk ke dalam satu alur kerja yang terintegrasi sehingga proses penyusunan katalog produk dapat dilakukan secara lebih efisien. Hasil analisis tersebut menjadi dasar dalam proses analisis kebutuhan sistem pada sub bab berikutnya.

15 3.3.2 Identifikasi Permasalahan

Berdasarkan hasil analisis proses bisnis yang telah dijelaskan pada Sub Bab 3.4.1, diketahui bahwa proses pengumpulan informasi produk dari platform 1688.com masih menghadapi beberapa kendala yang menyebabkan proses kerja menjadi kurang efisien. Kendala tersebut berkaitan dengan proses pengumpulan informasi, penerjemahan teks, pengolahan gambar produk, serta penyusunan katalog yang masih dilakukan secara manual. Oleh karena itu, identifikasi

permasalahan dilakukan sebagai dasar dalam merumuskan kebutuhan sistem yang akan dikembangkan.

Ringkasan hasil identifikasi permasalahan pada sistem berjalan disajikan pada Tabel 3.2.

Tabel 3.2 Analisis Permasalahan Sistem Berjalan

No	Permasalahan	Dampak
1	Informasi produk menggunakan bahasa Mandarin.	Pelaku usaha mengalami kesulitan dalam memahami spesifikasi dan informasi produk secara cepat.
2	Teks pada gambar produk belum dapat diterjemahkan dan diganti secara otomatis.	Pengguna harus melakukan OCR, menerjemahkan, dan mengganti teks pada gambar secara manual sehingga proses menjadi kurang efisien.
3	Belum terdapat sistem yang mengintegrasikan proses <i>web scraping</i> , OCR, penerjemahan, <i>image text swapping</i> , dan penyusunan informasi produk.	Pengguna harus berpindah antar aplikasi sehingga proses kerja menjadi kurang efisien dan meningkatkan risiko kesalahan dalam pengolahan informasi.
4	Penyusunan katalog produk masih dilakukan secara manual dalam pengubahan bahasa.	Membutuhkan waktu dan tenaga yang lebih banyak dalam menyusun katalog produk.

Berdasarkan hasil identifikasi permasalahan pada Tabel 3.2, dapat disimpulkan bahwa proses pengelolaan informasi produk pada platform 1688.com masih menghadapi beberapa kendala, seperti perbedaan bahasa, keterbatasan penerjemahan teks pada gambar, serta penggunaan beberapa aplikasi secara terpisah dalam proses *web scraping*, OCR, penerjemahan, dan penyusunan katalog. Kondisi tersebut menyebabkan proses kerja menjadi kurang efisien serta meningkatkan potensi kesalahan dalam pengolahan informasi. Oleh karena itu, diperlukan suatu sistem yang mampu mengintegrasikan seluruh proses tersebut dalam satu platform. Analisis kebutuhan sistem untuk memenuhi kebutuhan tersebut dibahas pada sub bab berikutnya.

3.3.3⁸ Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk merumuskan kebutuhan yang harus dipenuhi oleh sistem berdasarkan hasil analisis proses bisnis, identifikasi permasalahan, dan observasi terhadap platform 1688.com. Tahap ini bertujuan untuk memastikan bahwa sistem yang dikembangkan mampu membantu pelaku usaha dalam proses pengadaan data produk secara lebih efektif dan efisien.

Kebutuhan sistem pada penelitian ini diperoleh melalui observasi terhadap proses pengumpulan informasi produk, karakteristik platform 1688.com, serta kendala yang dihadapi oleh pelaku usaha selama proses pencarian, penerjemahan, dan penyusunan informasi produk. Berdasarkan hasil observasi dan analisis tersebut, kebutuhan sistem dikelompokkan menjadi kebutuhan fungsional dan kebutuhan non fungsional.

3.3.3.1 Analisis Kebutuhan Berdasarkan Observasi

Observasi dilakukan secara langsung terhadap platform 1688.com dengan mengamati alur penggunaan, struktur halaman produk, serta proses pengumpulan informasi produk yang dilakukan oleh pelaku usaha. Observasi juga dilakukan terhadap aktivitas penyusunan katalog produk untuk mengidentifikasi kebutuhan sistem yang sesuai dengan kondisi nyata pada proses bisnis.

Hasil observasi menunjukkan bahwa proses pengumpulan informasi produk masih menghadapi beberapa kendala, antara lain penggunaan bahasa Mandarin pada informasi produk, keberadaan informasi penting dalam bentuk gambar (*in-image text*), penggunaan beberapa aplikasi secara terpisah, serta penyusunan katalog produk yang masih dilakukan secara manual. Selain itu, halaman produk pada platform 1688.com memuat informasi secara dinamis sehingga diperlukan mekanisme yang mampu memperoleh data secara lengkap dari halaman yang telah dimuat pada browser.

Ringkasan hasil observasi dan kebutuhan sistem yang diperoleh disajikan pada Tabel 3.3.

Tabel 3.3 Kebutuhan Sistem

No	Hasil Observasi	Kebutuhan Sistem
1	Informasi produk masih menggunakan bahasa Mandarin.	Sistem mampu menerjemahkan informasi produk ke dalam bahasa Indonesia

No	Hasil Observasi	Kebutuhan Sistem
		sehingga lebih mudah dipahami oleh pelaku usaha.
2	Sebagian besar informasi penting berada pada gambar produk (<i>in-image text</i>).	Sistem mampu mendeteksi teks pada gambar, menerjemahkannya, serta menampilkan kembali hasil terjemahan pada gambar produk (<i>image text swapping</i>).
3	Halaman produk dimuat secara dinamis (<i>dynamic rendering</i>).	Sistem mampu melakukan <i>DOM scraping</i> pada halaman yang telah dimuat sehingga informasi produk dapat diperoleh secara lengkap.
4	Pengaturan parameter setiap tahapan pemrosesan memengaruhi efisiensi pipeline sistem.	Sistem mampu mengoptimalkan konfigurasi <i>pipeline</i> menggunakan Firefly Algorithm.
5	Pengumpulan informasi produk masih melibatkan beberapa aplikasi yang berbeda.	Sistem mampu mengintegrasikan proses <i>web scraping</i> , OCR, penerjemahan, <i>image text swapping</i> , penyusunan deskripsi produk, dan penyimpanan data dalam satu sistem.
6	Penyusunan katalog produk masih dilakukan secara manual.	Sistem mampu menyimpan hasil pengolahan informasi produk ke dalam basis data serta menyajikannya melalui <i>Web Dashboard</i> untuk mempermudah penyusunan katalog produk.

Berdasarkan hasil observasi tersebut, diperoleh kebutuhan sistem yang menjadi dasar dalam penyusunan kebutuhan fungsional dan kebutuhan non fungsional pada penelitian ini.

3.3.3.2 Kebutuhan Fungsional

Kebutuhan fungsional merupakan fungsi utama yang harus dimiliki oleh sistem agar dapat mendukung proses pengumpulan dan pengolahan informasi

produk sesuai dengan kebutuhan pengguna. Kebutuhan ini disusun berdasarkan hasil observasi dan identifikasi permasalahan pada sistem yang berjalan.

²
Tabel 3.4 Kebutuhan Fungsional Sistem

Kode	Kebutuhan Fungsional
F-01	Sistem mampu mengambil gambar produk dari halaman 1688.com secara otomatis.
F-02	Sistem mampu memperoleh informasi produk melalui <i>DOM scraping</i> pada halaman yang telah dimuat di browser.
F-03	Sistem mampu mendeteksi teks pada gambar produk menggunakan OCR.
F-04	Sistem mampu menerjemahkan hasil ekstraksi teks menggunakan Google Translate API.
F-05	Sistem mampu menampilkan hasil terjemahan kembali pada gambar produk (<i>image text swapping</i>).
F-06	Sistem mampu menghasilkan nama produk, deskripsi produk, dan kata kunci menggunakan Gemini.
F-07	Sistem mampu mengoptimalkan konfigurasi <i>Pipeline Scheduler</i> menggunakan Firefly Algorithm.
F-08	Sistem mampu menyimpan hasil pengolahan informasi produk ke dalam basis data.
F-09	Sistem mampu menampilkan informasi produk melalui Web Dashboard.
F-10	Sistem mampu melakukan pencarian dan penyaringan informasi produk.
F-11	Sistem mampu mengunduh hasil gambar terjemahan dan informasi produk.

Berdasarkan kebutuhan fungsional tersebut, sistem diharapkan mampu mengotomatisasi proses pengumpulan dan pengolahan informasi produk mulai dari proses *DOM scraping*, OCR, penerjemahan, *image text swapping*, optimasi *Pipeline Scheduler* menggunakan Firefly Algorithm, hingga penyajian informasi produk melalui Web Dashboard dalam satu sistem yang terintegrasi.

3.3.3.3 Kebutuhan Non-Fungsional

Selain kebutuhan fungsional, sistem juga memiliki kebutuhan non-fungsional yang berkaitan dengan kualitas sistem. Kebutuhan non-fungsional digunakan untuk memastikan sistem memiliki karakteristik operasional yang baik sehingga mampu berjalan secara andal, mudah digunakan, kompatibel dengan lingkungan implementasi, serta menjaga konsistensi data selama proses pengolahan informasi produk.

Tabel 3.5 Kebutuhan Non-Fungsional Sistem

Kode	Kebutuhan Non-Fungsional
NF-01	Sistem dapat dijalankan pada browser Google Chrome.
NF-02	Sistem memiliki antarmuka yang mudah digunakan (<i>user friendly</i>).
NF-03	Sistem mampu menjaga konsistensi dan integritas data selama proses pengolahan informasi produk berlangsung.
NF-04	Sistem mampu terhubung dengan layanan Google Translate API dan Gemini API selama proses penerjemahan dan penyusunan deskripsi produk.
NF-05	Sistem memerlukan koneksi internet selama proses <i>scraping</i> , penerjemahan, dan pengolahan data berlangsung.

3.4 Perancangan Sistem

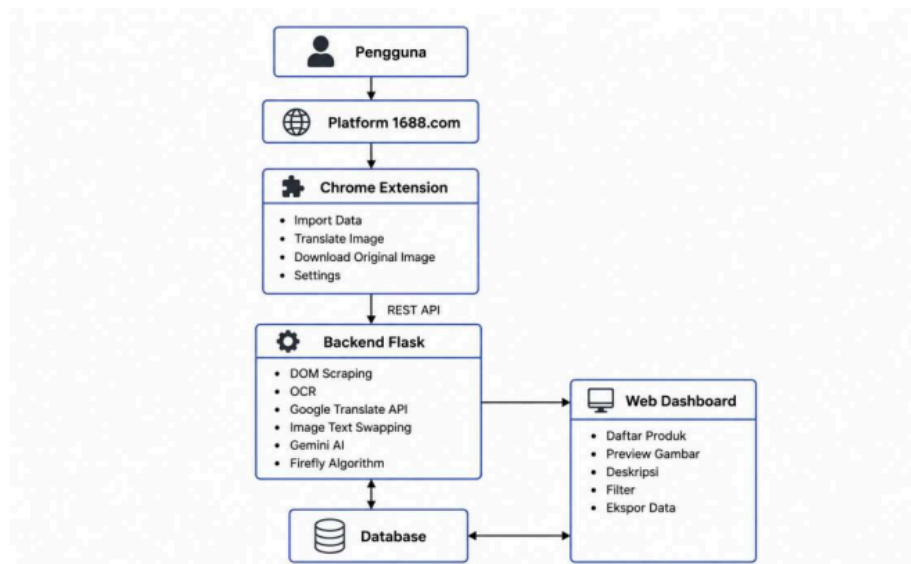
Berdasarkan hasil analisis sistem yang telah dilakukan, tahap selanjutnya adalah menyusun perancangan sistem sebagai acuan dalam proses implementasi. Perancangan sistem bertujuan untuk menghasilkan desain yang mampu memenuhi kebutuhan fungsional dan non-fungsional yang telah dirumuskan, sehingga sistem dapat mendukung proses pengadaan data produk dari platform 1688.com secara lebih efektif dan terintegrasi.

Perancangan sistem pada penelitian ini meliputi gambaran umum sistem, arsitektur sistem, pemodelan menggunakan *Unified Modeling Language* (UML), perancangan basis data, perancangan antarmuka pengguna, serta perancangan

modul sistem. Seluruh hasil perancangan tersebut menjadi dasar dalam proses implementasi sistem.

142 3.4.1 Gambaran Umum Sistem

Sistem yang diusulkan merupakan sistem pengumpulan dan pengolahan informasi produk berbasis Chrome Extension yang terintegrasi dengan Backend Flask, Database, dan Web Dashboard. Sistem ini dirancang untuk membantu pelaku usaha memperoleh informasi produk dari platform 1688.com secara lebih cepat dan efisien melalui proses pengambilan data produk, pengolahan informasi, serta pengelolaan hasil pemrosesan dalam satu sistem yang terintegrasi.



144 **Gambar 3.3** Gambaran Umum Sistem

Berdasarkan Gambar 3.3, proses sistem diawali ketika pengguna mengakses halaman produk pada platform 1688.com menggunakan peramban Google Chrome yang telah dipasang Chrome Extension. Chrome Extension berfungsi sebagai antarmuka utama yang menyediakan fitur Import Data, Translate Image, Download Original Image, Settings, dan akses menuju Web Dashboard.

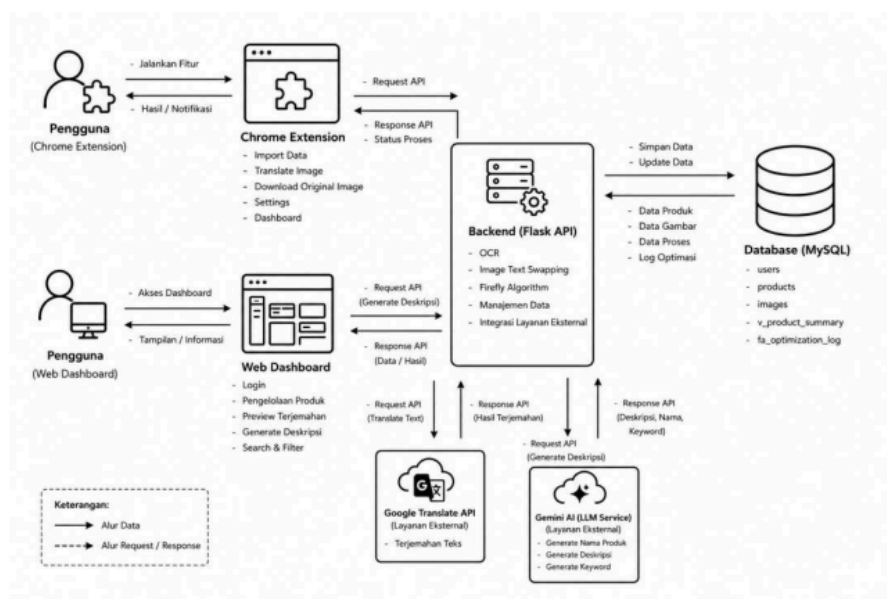
Setiap permintaan yang dilakukan melalui Chrome Extension dikirimkan ke Backend Flask melalui REST API untuk diproses. Backend Flask mengimplementasikan berbagai modul utama, yaitu DOM Scraping, Optical Character Recognition (OCR), Google Translate API, Image Text Swapping, Gemini AI, dan Firefly Algorithm. Seluruh hasil pemrosesan kemudian disimpan ke dalam Database sebagai media penyimpanan data produk dan hasil pengolahan.

Selanjutnya, data yang telah tersimpan pada Database dapat diakses melalui Web Dashboard untuk mendukung pengelolaan informasi produk. Dashboard menyediakan berbagai fungsi, seperti menampilkan daftar produk, meninjau hasil terjemahan gambar, melihat informasi produk yang dihasilkan sistem, melakukan pencarian dan penyaringan data, serta mengekspor data sesuai kebutuhan pengguna.

Secara umum, sistem yang diusulkan mengintegrasikan Chrome Extension, Backend Flask, Database, dan Web Dashboard ke dalam satu arsitektur yang saling terhubung. Integrasi tersebut memungkinkan proses pengambilan data, pengolahan informasi produk, penyimpanan data, serta pengelolaan hasil pemrosesan dilakukan secara lebih efisien dibandingkan proses manual.

3.5 Arsitektur Sistem Usulan

Arsitektur sistem usulan menggambarkan hubungan antar komponen yang membentuk sistem pengumpulan dan pengolahan informasi produk berbasis Chrome Extension pada platform 1688.com. Arsitektur ini menggunakan pendekatan client-server, di mana Chrome Extension berperan sebagai *client* yang berinteraksi langsung dengan halaman produk, sedangkan Backend Flask berfungsi sebagai pusat pemrosesan dan integrasi layanan sistem. Backend mengelola proses OCR, penerjemahan menggunakan Google Translate API, *Image Text Swapping*, integrasi dengan Gemini AI, serta optimasi pipeline menggunakan Firefly Algorithm. Seluruh hasil pemrosesan disimpan pada basis data MySQL dan dikelola melalui Web Dashboard. Arsitektur sistem usulan ditunjukkan pada Gambar 3.4.



Gambar 3.4 Arsitektur Sistem Usulan

Berdasarkan Gambar 3.4, proses sistem diawali ketika pengguna membuka halaman produk pada platform 1688.com menggunakan Google Chrome yang telah dipasang Chrome Extension. Melalui ekstensi tersebut, pengguna dapat menjalankan fitur Import Data, Translate Image, Download Original Image, Settings, serta mengakses Web Dashboard. Setiap permintaan dari Chrome Extension dikirimkan ke Backend Flask melalui REST API untuk diproses lebih lanjut.

Backend Flask berperan sebagai pusat pemrosesan sistem. Data yang diterima dari Chrome Extension diproses melalui beberapa tahapan, yaitu ekstraksi teks menggunakan Optical Character Recognition (OCR), penerjemahan menggunakan Google Translate API, serta penggantian teks pada gambar melalui mekanisme Image Text Swapping. Selain itu, Firefly Algorithm digunakan untuk mengoptimalkan konfigurasi pipeline pemrosesan sehingga proses pengolahan data dapat berlangsung secara lebih efisien.

Pada Web Dashboard, pengguna dapat mengelola data produk, melihat hasil terjemahan gambar, melakukan pencarian data, serta menghasilkan deskripsi melalui fitur Generate Deskripsi. Ketika fitur tersebut dijalankan, Web Dashboard mengirimkan permintaan ke Backend Flask, kemudian Backend berkomunikasi dengan Gemini AI untuk menghasilkan informasi produk berdasarkan data yang telah tersimpan pada basis data. Hasil yang diperoleh selanjutnya disimpan kembali ke dalam basis data dan ditampilkan pada Web Dashboard.

Seluruh data sistem disimpan pada Database MySQL yang terdiri atas tabel users, products, dan images, serta objek v_product_summary dan fa_optimization_log. Basis data digunakan untuk menyimpan informasi pengguna, data produk, hasil pengolahan gambar, ringkasan informasi produk, serta log proses optimasi Firefly Algorithm sehingga seluruh komponen sistem dapat mengakses data secara terpusat dan konsisten.

Dengan arsitektur tersebut, proses pengambilan data produk, pengolahan gambar, penyimpanan data, serta pengelolaan informasi produk dapat dilakukan secara terintegrasi melalui Chrome Extension, Backend Flask, Database MySQL, dan Web Dashboard. Pendekatan ini memungkinkan proses penyusunan katalog produk menjadi lebih efisien dibandingkan proses manual karena seluruh tahapan pengolahan dilakukan dalam satu sistem yang saling terhubung.

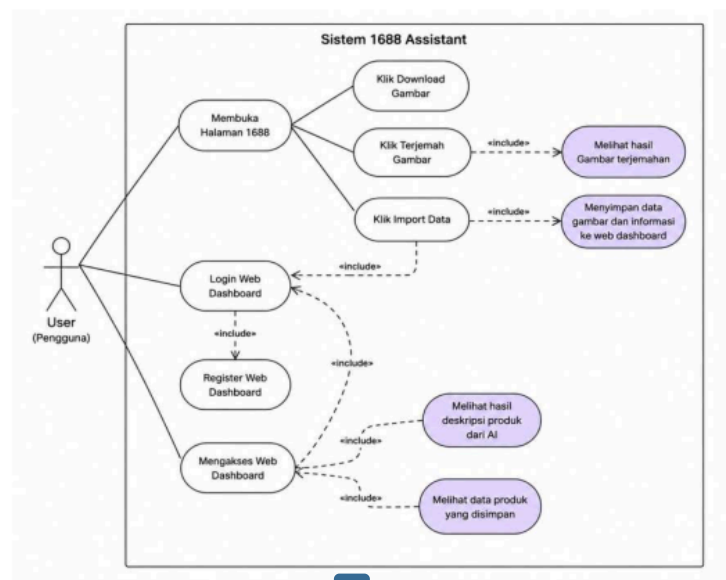
119

3.5.1 Use Case Diagram

Use Case Diagram digunakan untuk menggambarkan hubungan antara aktor dengan fungsi-fungsi yang disediakan oleh sistem. Diagram ini menunjukkan

layanan yang dapat diakses oleh pengguna tanpa menjelaskan proses internal sistem.

Pada sistem yang diusulkan, terdapat satu aktor utama, yaitu Pelaku Usaha, yang menggunakan sistem untuk memperoleh dan mengelola informasi produk dari platform 1688.com melalui Chrome Extension dan Web Dashboard. Melalui Chrome Extension, pelaku usaha dapat melakukan pengambilan informasi produk (*DOM scraping*), menerjemahkan teks pada gambar produk menggunakan OCR dan Google Translate API, melakukan *image text swapping*, mengunduh gambar asli produk, serta menghasilkan nama produk, deskripsi produk, dan kata kunci menggunakan Gemini. Selain itu, melalui Web Dashboard pengguna dapat mengelola dan meninjau data produk yang telah diproses oleh sistem.



Gambar 3.5 ⁹² Use Case Diagram

Dengan adanya *use case diagram* tersebut, dapat diketahui bahwa seluruh fungsi sistem berpusat pada kebutuhan pelaku usaha dalam memperoleh dan mengelola informasi produk secara otomatis dari platform 1688.com.

¹ 3.5.2 Activity Diagram

Activity Diagram digunakan untuk menggambarkan alur aktivitas yang terjadi pada sistem ketika pengguna menjalankan suatu fitur hingga sistem menghasilkan keluaran. Diagram ini menunjukkan urutan aktivitas, percabangan keputusan (*decision*), serta interaksi antara pengguna, Ekstensi Chrome, Backend Flask, dan Database MySQL dalam menjalankan proses bisnis sistem.

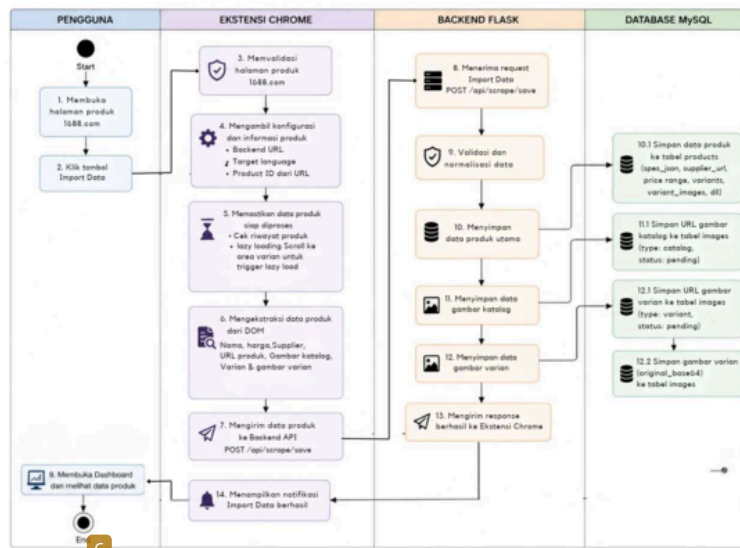
Pada penelitian ini, Activity Diagram disusun berdasarkan fitur-fitur utama yang tersedia pada sistem. Pendekatan ini dipilih agar setiap proses bisnis dapat dijelaskan secara lebih spesifik sesuai dengan implementasi masing-masing fitur. Activity Diagram yang disajikan meliputi proses Import Data Produk, Terjemah Gambar, Download Gambar, Generate Deskripsi Produk, dan Pengaturan Bahasa (Settings).

Pada penelitian ini, Activity Diagram difokuskan pada dua proses utama, yaitu proses pengambilan data produk dari platform 1688.com dan proses OCR serta translasi gambar produk.

a. Activity Diagram Fitur Import Data Produk

Activity Diagram ini menggambarkan alur proses penyimpanan data produk dari platform 1688.com ke dalam sistem. Proses diawali ketika pengguna membuka halaman detail produk pada platform 1688.com menggunakan Google Chrome yang telah dipasang ekstensi. Setelah pengguna menekan tombol Import Data, ekstensi melakukan proses DOM Scraping untuk mengambil informasi produk, kemudian mengirimkan data tersebut ke Backend Flask melalui REST API.

Backend menerima dan memvalidasi data yang dikirimkan. Apabila data valid, backend melakukan normalisasi informasi produk, menyimpan data produk ke basis data, serta menyimpan informasi gambar yang berkaitan dengan produk. Setelah seluruh proses penyimpanan selesai, backend mengirimkan respons berhasil kepada ekstensi sehingga pengguna memperoleh notifikasi bahwa data produk berhasil diimpor ke dalam sistem.

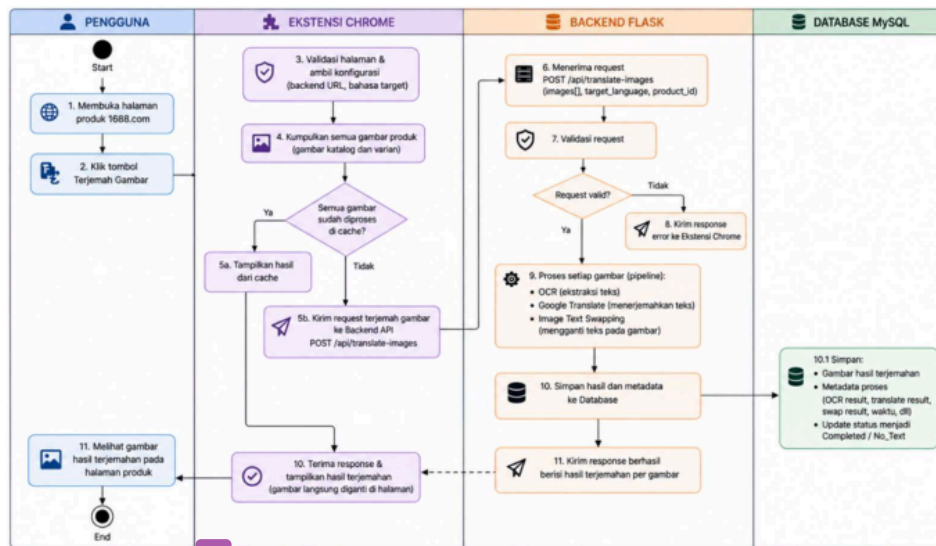


Gambar 3.6 Activity Diagram Fitur Import Data Produk

b. Activity Diagram Fitur Terjemah Gambar

Activity Diagram ini menggambarkan proses penerjemahan gambar produk yang mengandung teks berbahasa Mandarin. Proses dimulai ketika pengguna menekan tombol Terjemah Gambar pada ekstensi. Ekstensi kemudian mengirimkan daftar gambar produk ke Backend Flask untuk diproses.

Backend menjalankan tahapan OCR untuk mengekstraksi teks dari gambar, menerjemahkan hasil ekstraksi menggunakan Google Translate API, kemudian melakukan proses *Image Text Swapping* untuk mengganti teks asli pada gambar dengan hasil terjemahan. Setelah proses selesai, gambar hasil terjemahan beserta metadata disimpan ke dalam basis data. Selanjutnya backend mengirimkan respons kepada ekstensi sehingga gambar hasil terjemahan dapat ditampilkan kepada pengguna.

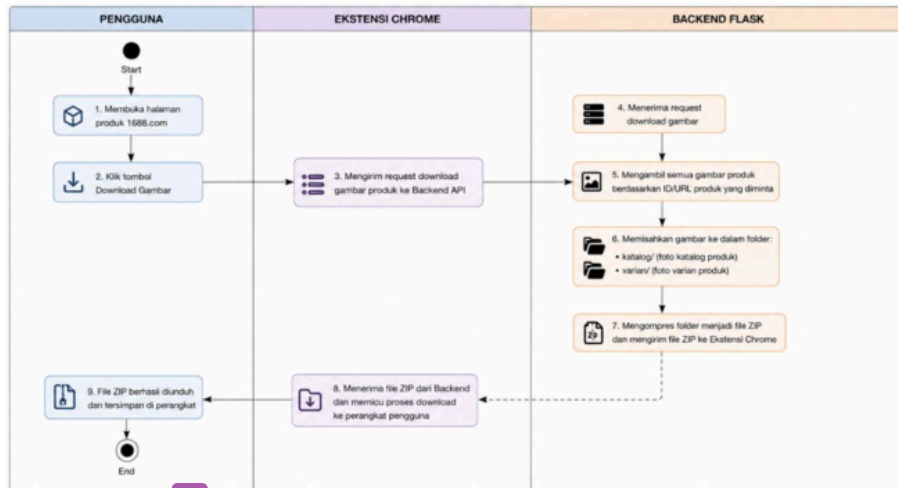


Gambar 3.7 Activity Diagram Fitur Terjemah Gambar

c. Activity Diagram Fitur Download Gambar

Activity Diagram ini menggambarkan proses pengunduhan gambar produk. Proses diawali ketika pengguna memilih fitur Download Gambar pada ekstensi. Selanjutnya ekstensi mengirimkan permintaan pengunduhan ke Backend Flask.

Backend mengumpulkan seluruh gambar produk berdasarkan URL yang diterima, memisahkan gambar katalog dan gambar varian, kemudian mengompres seluruh gambar ke dalam berkas ZIP. Setelah proses kompresi selesai, backend mengirimkan berkas ZIP kepada ekstensi sehingga browser dapat memulai proses pengunduhan dan menyimpan file ke perangkat pengguna.

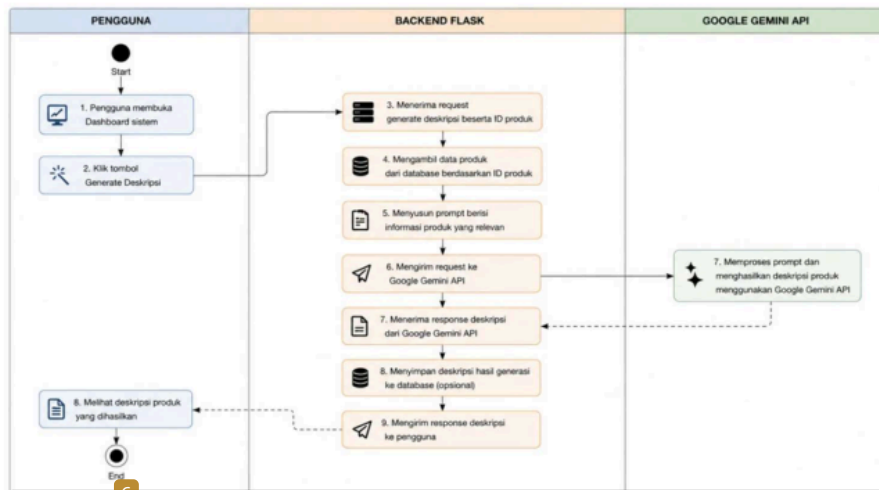


57
Gambar 3.8 Activity Diagram Fitur Download Gambar

d. Activity Diagram Fitur Generate Deskripsi Produk

Activity Diagram ini menggambarkan proses pembuatan deskripsi produk menggunakan layanan Gemini AI. Proses dimulai ketika pengguna memilih salah satu produk pada Web Dashboard dan menekan tombol Generate Deskripsi.

Backend mengambil informasi produk dari basis data kemudian mengirimkan data tersebut ke layanan Gemini AI untuk menghasilkan nama produk, deskripsi, dan kata kunci. Hasil yang diperoleh selanjutnya disimpan kembali ke dalam basis data dan ditampilkan pada Web Dashboard sehingga dapat digunakan sebagai informasi tambahan dalam penyusunan katalog produk.

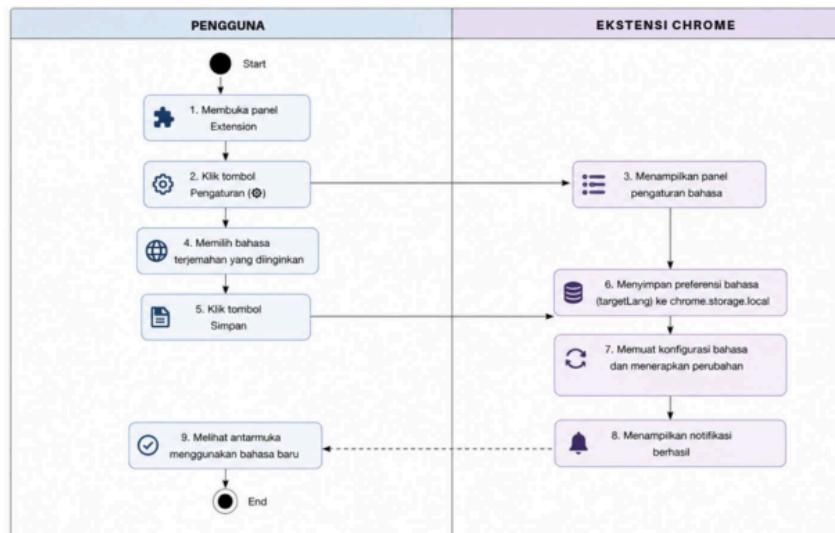


6
Gambar 3.9 Activity Diagram Fitur Generate Deskripsi Produk

e. **Activity Diagram** Fitur Pengaturan Bahasa (Settings)

Activity Diagram ini menggambarkan proses pengaturan bahasa tujuan pada ekstensi. Pengguna membuka menu Settings, kemudian memilih bahasa yang diinginkan sebagai bahasa target penerjemahan.

Setelah pengguna menekan tombol Simpan, konfigurasi bahasa disimpan pada penyimpanan lokal (*Chrome Storage*). Selanjutnya antarmuka ekstensi diperbarui secara otomatis sesuai dengan bahasa yang dipilih sehingga seluruh proses penerjemahan berikutnya menggunakan konfigurasi bahasa tersebut.



Gambar 3.10 *Activity Diagram* Fitur Pengaturan Bahasa (Settings)

3.5.3 **Sequence Diagram**

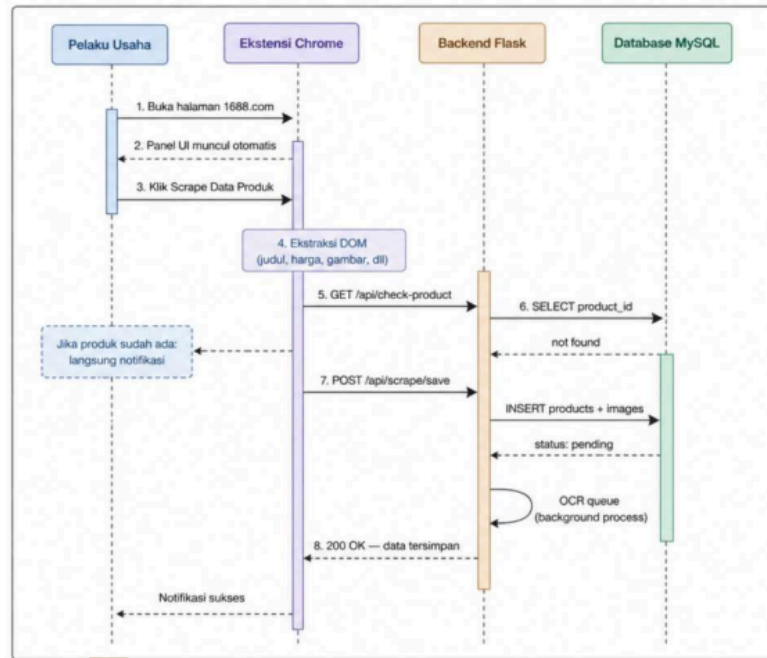
Sequence Diagram digunakan untuk menggambarkan urutan komunikasi antar objek berdasarkan dimensi waktu selama menjalankan suatu proses. Diagram ini memperlihatkan pesan (*message*) yang dikirim antar komponen sistem sehingga hubungan antar komponen dapat dipahami secara lebih jelas.

Pada penelitian ini, *Sequence Diagram* digunakan untuk memodelkan dua proses utama, yaitu komunikasi selama proses pengambilan data produk serta komunikasi selama proses OCR dan translasi gambar.

a. **Sequence Diagram** Proses Pengambilan Data Produk

Sequence Diagram ini menggambarkan interaksi antara pelaku usaha, ekstensi Google Chrome, backend Flask, dan basis data selama proses scraping berlangsung. Proses dimulai ketika pengguna membuka halaman produk dan menjalankan proses scraping melalui ekstensi. Ekstensi kemudian mengekstraksi

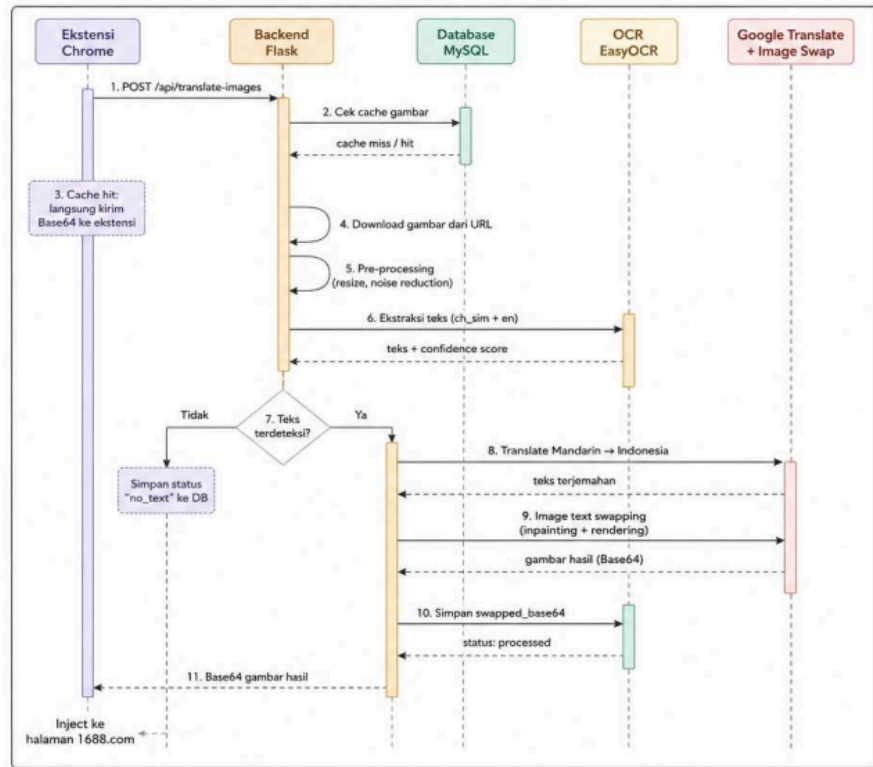
data dari DOM dan mengirimkannya ke backend untuk dilakukan validasi dan penyimpanan data. Setelah proses penyimpanan selesai, backend mengirimkan respons ke ekstensi sehingga pengguna memperoleh notifikasi bahwa proses berhasil dilakukan. Pipeline *OCR* dijalankan sebagai proses latar belakang setelah data berhasil disimpan.



150
Gambar 3.11 *Sequence Diagram* pengambilan data produk

b. *Sequence Diagram* Proses OCR dan Translasi

Sequence Diagram ini menggambarkan komunikasi antara backend Flask, basis data, modul OCR, dan layanan translasi selama proses pengolahan gambar berlangsung. Backend terlebih dahulu memeriksa ketersediaan hasil pada cache. Apabila hasil belum tersedia, backend mengunduh gambar, melakukan *pre-processing*, menjalankan OCR menggunakan EasyOCR, menerjemahkan teks hasil ekstraksi, kemudian melakukan proses image text swapping. Hasil akhir berupa gambar yang telah diterjemahkan disimpan ke dalam basis data dan dikirimkan kembali ke ekstensi untuk ditampilkan pada halaman produk.



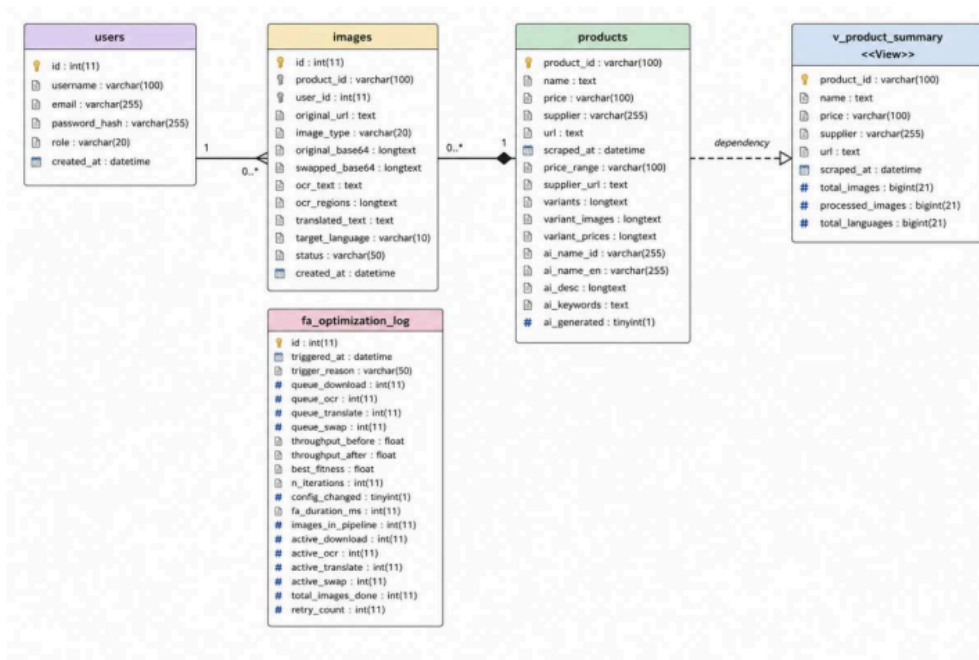
Gambar 3.12 *Sequence Diagram* OCR dan Translasi

3.5.4 ¹ Perancangan Basis Data

Perancangan basis data dilakukan untuk mendukung proses penyimpanan, pengelolaan, dan pengambilan data yang dihasilkan selama sistem berjalan. Basis data pada penelitian ini menggunakan MySQL sebagai *Database Management System* (DBMS) karena mampu mengelola data relasional secara efisien, mendukung integrasi dengan framework Flask, serta memiliki performa yang baik dalam menangani proses penyimpanan dan pengambilan data.

Basis data dirancang untuk menyimpan informasi produk hasil pengambilan data dari platform 1688.com, hasil pengolahan gambar menggunakan *Optical Character Recognition* (OCR), hasil penerjemahan menggunakan Google Translate API, hasil *Image Text Swapping*, informasi produk yang dihasilkan melalui Gemini AI, data pengguna, serta log proses optimasi Firefly Algorithm. Struktur basis data disusun agar setiap informasi saling terhubung sehingga memudahkan proses pengelolaan data, menjaga konsistensi informasi, serta meminimalkan redundansi data.

Perancangan struktur basis data direpresentasikan menggunakan ⁹³ *Class Diagram* yang ditunjukkan pada Gambar 3.13.



Gambar 3.13 Class Diagram Database katalog_1688

Berdasarkan Class Diagram yang telah dirancang, basis data terdiri atas lima kelas utama, yaitu **users**, **products**, **images**, **v_product_summary**, dan **fa_optimization_log**. Kelas **users** digunakan untuk menyimpan informasi pengguna yang memiliki hak akses terhadap sistem. Kelas **products** berfungsi sebagai penyimpanan utama data produk hasil pengambilan data dari platform 1688.com, meliputi nama produk, harga, pemasok, variasi produk, serta informasi produk yang dihasilkan melalui Gemini AI, seperti nama produk, deskripsi, dan kata kunci.

Kelas **images** digunakan untuk menyimpan informasi gambar produk, meliputi URL gambar asli, hasil OCR, hasil penerjemahan, hasil *Image Text Swapping*, serta status proses pengolahan gambar. Setiap produk dapat memiliki lebih dari satu gambar sehingga hubungan antara kelas **products** dan **images** bersifat one-to-many. Selain itu, setiap data gambar dikaitkan dengan pengguna melalui atribut **user_id**, sehingga satu pengguna dapat memiliki banyak data gambar hasil pengolahan.

Sistem juga memanfaatkan **v_product_summary** sebagai *database view* yang digunakan untuk menyajikan ringkasan informasi produk, seperti jumlah gambar, jumlah gambar yang telah diproses, serta jumlah bahasa hasil penerjemahan. Informasi tersebut dimanfaatkan untuk mendukung penyajian data pada Web Dashboard tanpa perlu melakukan proses agregasi secara berulang.

Selain itu, kelas `fa_optimization_log` digunakan untuk menyimpan riwayat proses optimasi yang dilakukan oleh Firefly Algorithm, meliputi penyebab optimasi dijalankan, konfigurasi pipeline sebelum dan sesudah optimasi, nilai *fitness*, serta informasi performa proses. Data tersebut digunakan sebagai dokumentasi sekaligus bahan evaluasi terhadap kinerja optimasi pipeline yang diterapkan pada sistem.

Perancangan hubungan antar kelas dilakukan untuk menjaga integritas data, mengurangi redundansi penyimpanan, serta mendukung proses pengolahan, penyajian informasi produk, dan evaluasi kinerja sistem secara efisien.

3.5.5 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk merancang media interaksi antara pelaku usaha dengan sistem sehingga seluruh fungsi dapat digunakan secara mudah, efisien, dan intuitif. Antarmuka sistem dirancang dengan mempertimbangkan alur kerja pengadaan produk dari platform 1688.com, sehingga pengguna dapat melakukan proses DOM scraping, translasi gambar, pengelolaan data produk, serta pemanfaatan fitur berbasis Artificial Intelligence melalui tampilan yang sederhana dan mudah dipahami.

Pada penelitian ini, antarmuka sistem terdiri atas dua komponen utama, yaitu ekstensi Google Chrome dan Web Dashboard. Ekstensi Google Chrome berfungsi sebagai media untuk melakukan proses DOM scraping, menerjemahkan teks pada gambar produk melalui proses OCR, Google Translate API, dan image text swapping, serta mengirimkan data hasil pengolahan ke backend. Sementara itu, Web Dashboard berfungsi sebagai media untuk mengelola data produk hasil scraping, menampilkan informasi produk yang telah diproses, serta menyediakan fasilitas pencarian dan pengelolaan data produk.

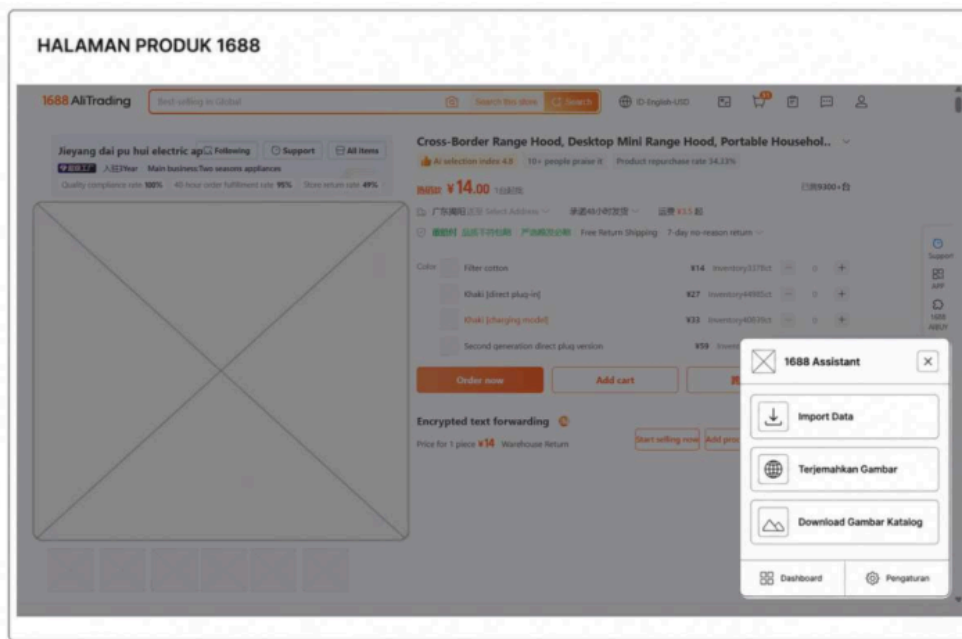
3.5.5.1 Perancangan Antarmuka Ekstensi Google Chrome

Perancangan antarmuka ekstensi Google Chrome bertujuan untuk menyediakan media interaksi bagi pengguna dalam melakukan proses pengambilan dan pengolahan data produk secara langsung pada halaman produk 1688.com. Antarmuka ekstensi dirancang agar muncul setelah pengguna mengaktifkan ekstensi pada halaman produk, sehingga seluruh fungsi utama dapat diakses tanpa harus berpindah ke aplikasi lain.

Antarmuka ekstensi menyediakan tiga fitur utama, yaitu Import Data untuk mengambil informasi produk menggunakan teknik DOM scraping, Terjemahkan Gambar untuk menerjemahkan teks berbahasa Mandarin pada gambar melalui proses OCR, Google Translate API, dan image text swapping, serta Download Gambar Katalog untuk mengunduh seluruh gambar produk yang tersedia pada

halaman. Selain itu, ekstensi juga menyediakan akses menuju Dashboard dan Pengaturan sebagai navigasi tambahan bagi pengguna.

Rancangan antarmuka pada Gambar 3.14 memperlihatkan posisi ekstensi yang ditampilkan di sisi kanan halaman produk 1688.com. Penempatan tersebut memungkinkan pengguna menjalankan seluruh proses pengolahan data secara langsung tanpa mengganggu aktivitas penelusuran informasi produk.



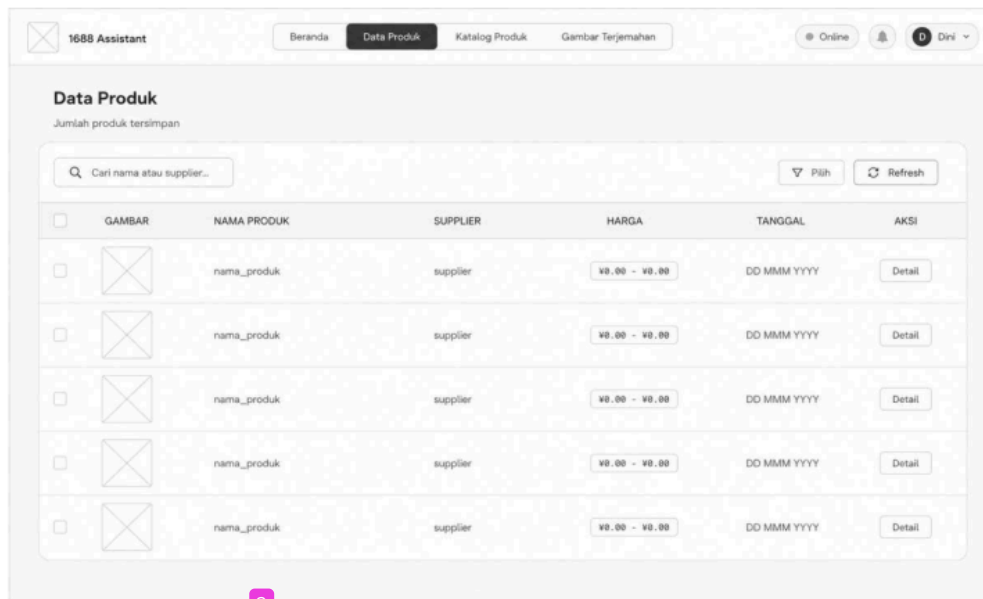
Gambar 3.14 Perancangan Antarmuka Ekstensi Google Chrome

3.5.5.2 Perancangan Antarmuka Dashboard

Web Dashboard dirancang sebagai pusat pengelolaan seluruh data hasil *DOM scraping* yang dikirimkan dari ekstensi Google Chrome. Melalui dashboard, pengguna dapat melihat daftar produk yang telah diproses, melakukan pencarian produk berdasarkan nama atau kata kunci tertentu, serta memantau status pemrosesan setiap produk.

Setiap data produk ditampilkan dalam bentuk tabel yang memuat gambar utama, nama produk, status pemrosesan, tanggal pengolahan, dan aksi untuk melihat informasi produk secara lebih lengkap. Selain itu, dashboard juga menampilkan ringkasan jumlah produk yang telah diproses sehingga pengguna dapat memantau hasil pengolahan secara keseluruhan.

Perancangan antarmuka dashboard dilakukan dengan mengutamakan kemudahan navigasi dan penyajian informasi secara terstruktur sehingga pengguna dapat mengakses hasil pengolahan produk dengan lebih efisien.



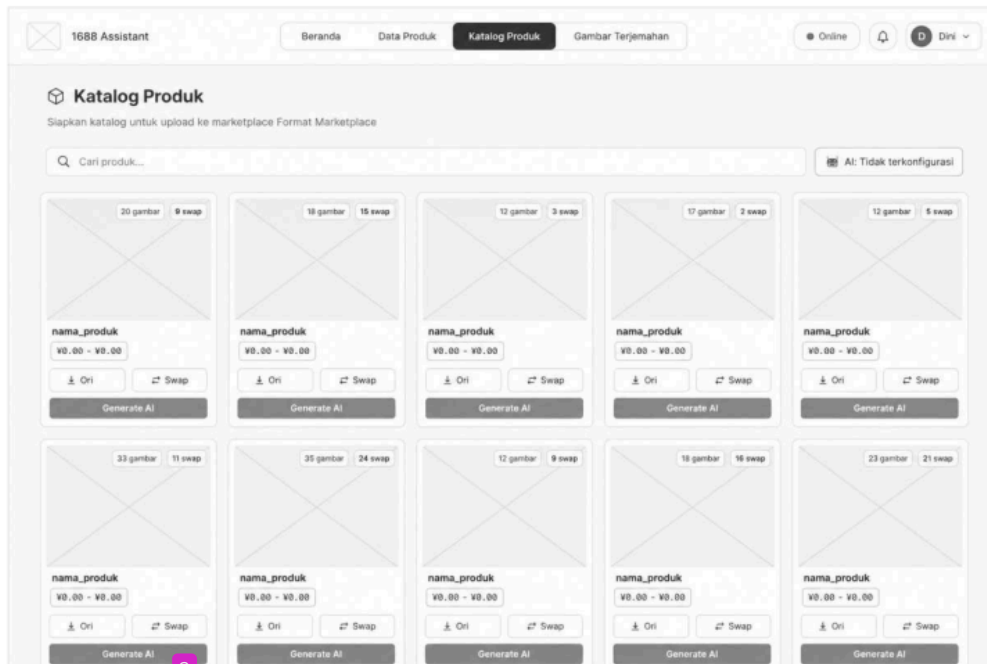
2 Gambar 3.15 Perancangan Antarmuka Dashboard

3.5.5.3 Perancangan Antarmuka Halaman Detail Produk

Halaman Detail Produk dirancang untuk menampilkan informasi hasil pengolahan dari satu produk yang dipilih pada menu Data Produk. Halaman ini menyediakan informasi produk yang telah dihasilkan oleh sistem, meliputi nama produk, deskripsi produk, dan kata kunci hasil pemrosesan menggunakan Gemini.

Selain informasi tekstual, halaman ini juga menampilkan galeri gambar yang berisi gambar asli (*original image*) dan gambar hasil *image text swapping* sehingga pengguna dapat membandingkan hasil penerjemahan secara langsung. Halaman ini juga menyajikan status setiap tahapan pengolahan, mulai dari proses *scraping*, OCR, penerjemahan, hingga *image text swapping*, sebagai informasi bahwa seluruh proses telah berhasil diselesaikan.

Perancangan halaman ini bertujuan agar seluruh hasil pengolahan produk dapat diakses dalam satu halaman sehingga memudahkan pengguna melakukan verifikasi dan penyusunan katalog produk multibahasa.



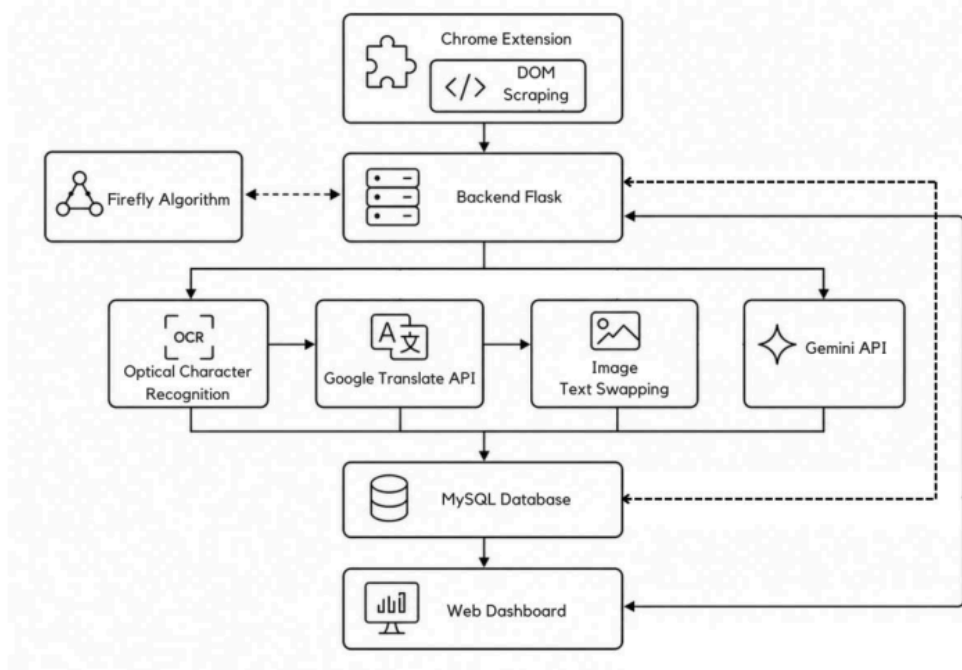
Gambar 3.16 Perancangan Antarmuka Halaman Detail Produk

3.6 Perancangan Modul Sistem

Perancangan modul sistem dilakukan untuk membagi sistem ke dalam beberapa komponen berdasarkan fungsi dan tanggung jawab masing-masing. Pendekatan modular diterapkan agar setiap komponen dapat dikembangkan, dipelihara, dan diuji secara independen tanpa memengaruhi modul lainnya. Selain itu, pembagian modul mempermudah proses integrasi antar komponen melalui mekanisme komunikasi yang terstruktur sehingga aliran data antar modul menjadi lebih jelas.

Pada sistem yang diusulkan, modul utama terdiri atas Chrome Extension, Backend Flask, Modul OCR, Modul Terjemahan, Modul Image Text Swapping, Gemini API, MySQL Database, Web Dashboard, dan Firefly Algorithm. Masing-masing modul memiliki fungsi yang saling melengkapi dalam membentuk alur pemrosesan sistem, mulai dari proses DOM scraping, ekstraksi teks menggunakan OCR, penerjemahan teks, penggantian teks pada gambar (*image text swapping*), optimasi penjadwalan proses menggunakan Firefly Algorithm, hingga penyimpanan dan penyajian hasil melalui Web Dashboard.

Hubungan antar modul sistem ditunjukkan pada Gambar 3.17.



Gambar 3.17 Hubungan Antar Modul Sistem

Berdasarkan **Gambar 3.17**, setiap **modul** pada sistem memiliki hubungan yang saling terintegrasi dalam mendukung proses pengolahan data. Chrome Extension berkomunikasi dengan Backend Flask untuk mengirimkan data hasil *DOM scraping* sekaligus menerima respons dari backend. Backend Flask berperan sebagai pusat koordinasi yang menghubungkan seluruh modul sehingga pertukaran data antar komponen dapat berlangsung secara terstruktur.

Dalam menjalankan proses pengolahan gambar, Backend Flask terhubung dengan Modul Optical Character Recognition (OCR), Google Translate API, dan Modul Image Text Swapping. Ketiga modul tersebut saling berhubungan membentuk rangkaian pengolahan gambar, di mana hasil dari suatu modul menjadi masukan bagi modul berikutnya. Selain itu, Backend Flask juga terhubung dengan Gemini API untuk menyediakan layanan pembuatan ringkasan informasi produk.

Hubungan antara Firefly Algorithm dan Backend Flask bersifat dua arah. Firefly Algorithm memberikan hasil optimasi parameter penjadwalan proses kepada Backend Flask, sedangkan Backend Flask menyediakan informasi yang diperlukan sebagai dasar evaluasi proses optimasi. Dengan demikian, keputusan penjadwalan yang dihasilkan dapat diterapkan pada proses pengolahan data di dalam sistem.

Seluruh hasil pemrosesan dari berbagai modul disimpan pada MySQL Database sebagai media penyimpanan terpusat. Selanjutnya, Web Dashboard

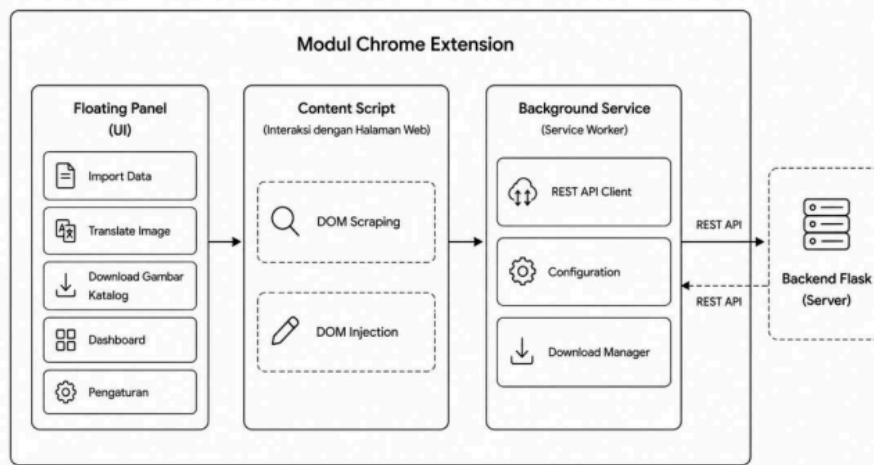
terhubung dengan MySQL Database untuk menampilkan data kepada pengguna. Selain itu, Web Dashboard juga berkomunikasi langsung dengan Backend Flask ketika pengguna menjalankan suatu fungsi yang memerlukan pemrosesan sistem, sehingga Backend Flask dapat mengakses maupun memperbarui data pada database sesuai kebutuhan.

3.6.1 Modul Chrome Extension

Modul Chrome Extension merupakan komponen *client-side* yang dirancang sebagai media interaksi antara pengguna dengan sistem pada platform 1688.com. Modul ini memungkinkan pengguna mengakses layanan sistem secara langsung melalui halaman produk tanpa perlu berpindah ke aplikasi lain. Selain berfungsi sebagai antarmuka pengguna, modul ini juga dirancang untuk menghubungkan halaman web dengan Backend Flask sehingga proses pengiriman data dan penerimaan hasil pemrosesan dapat dilakukan secara terintegrasi.

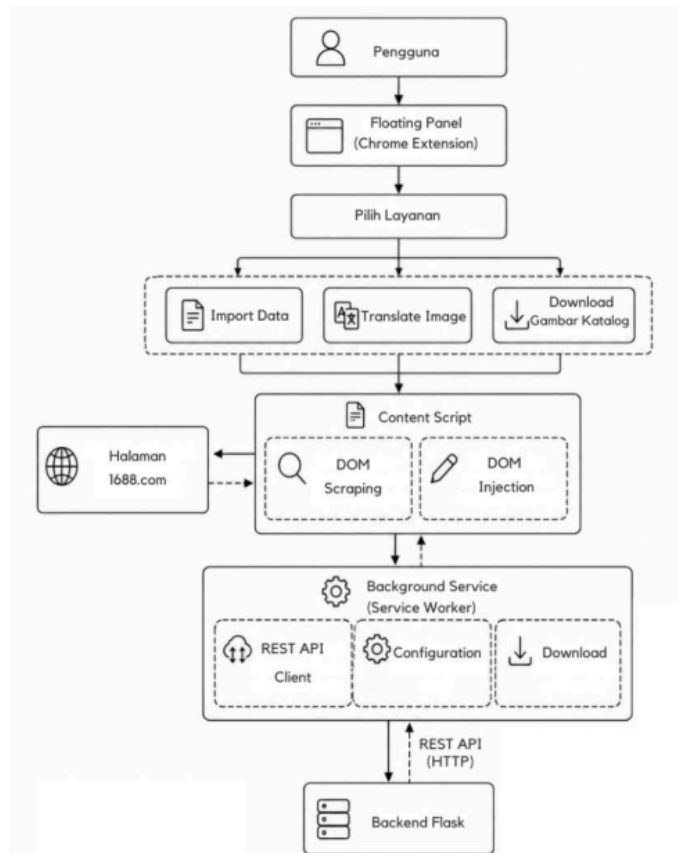
Secara umum, Modul Chrome Extension terdiri atas tiga komponen utama, yaitu Floating Panel, Content Script, dan Background Service. Floating Panel berfungsi sebagai antarmuka pengguna yang menyediakan akses terhadap layanan sistem. Content Script bertugas berinteraksi secara langsung dengan struktur halaman web (*Document Object Model* atau DOM), sedangkan Background Service bertugas mengelola komunikasi antara Chrome Extension dengan Backend Flask melalui REST API. Pembagian komponen tersebut dirancang agar setiap komponen memiliki tanggung jawab yang jelas sehingga proses interaksi pengguna, komunikasi sistem, dan pengelolaan data dapat dilakukan secara terpisah namun tetap saling terintegrasi.

Pada perancangannya, Content Script memanfaatkan mekanisme DOM Scraping dan DOM Injection sebagai bagian dari proses interaksi dengan halaman web. DOM Scraping digunakan untuk memperoleh informasi produk dengan mengakses elemen-elemen HTML pada halaman 1688.com, sedangkan DOM Injection digunakan untuk menampilkan kembali hasil pemrosesan yang diterima dari Backend Flask ke dalam halaman web. Dengan mekanisme tersebut, proses pengambilan data maupun penyajian hasil dapat dilakukan secara langsung pada halaman produk tanpa mengubah struktur utama halaman.



Gambar 3.18 Perancangan Modul Chrome Extension

Gambar 3.18 menunjukkan perancangan Modul Chrome Extension yang terdiri atas tiga komponen utama, yaitu Floating Panel, Content Script, dan Background Service. Floating Panel menyediakan antarmuka bagi pengguna untuk mengakses layanan sistem, seperti Import Data, Translate Image, Download Gambar Katalog, Dashboard, dan Pengaturan. Permintaan dari pengguna kemudian diteruskan ke Content Script untuk berinteraksi dengan halaman web melalui mekanisme DOM Scraping dan DOM Injection. Selanjutnya, Background Service berperan sebagai penghubung antara Chrome Extension dan Backend Flask melalui komunikasi REST API, sehingga proses pertukaran data antara *client-side* dan *server-side* dapat dilakukan secara terintegrasi.



Gambar 3.19 Alur Kerja Modul Chrome Extension

Gambar 3.19 menunjukkan alur kerja Modul Chrome Extension. Proses dimulai ketika pengguna memilih salah satu layanan melalui Floating Panel. Permintaan tersebut diteruskan ke Content Script, yang kemudian berinteraksi dengan halaman web menggunakan mekanisme DOM Scraping untuk memperoleh data yang dibutuhkan. Data hasil scraping selanjutnya dikirimkan ke Background Service, kemudian diteruskan ke Backend Flask melalui REST API untuk diproses sesuai dengan layanan yang dipilih pengguna. Setelah proses pada Backend Flask selesai, hasil pemrosesan dikirim kembali ke Background Service dan diteruskan ke Content Script. Selanjutnya, DOM Injection digunakan untuk memperbarui elemen pada halaman 1688.com sehingga hasil pemrosesan dapat ditampilkan secara langsung kepada pengguna.

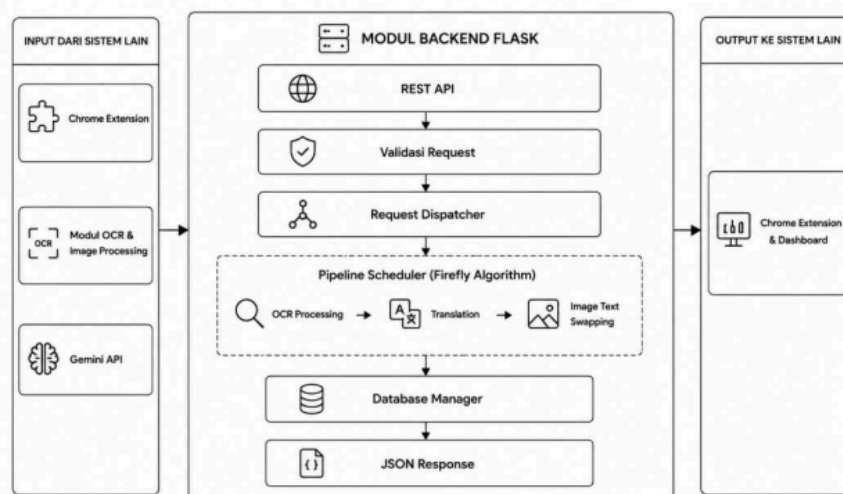
3.6.2 Modul Backend Flask

Modul Backend Flask berfungsi sebagai layanan backend yang menerima permintaan dari Chrome Extension maupun Web Dashboard, kemudian mengkoordinasikan seluruh proses pengolahan data pada sistem. Backend menyediakan layanan REST API sebagai media komunikasi antar komponen

sistem, melakukan validasi data yang diterima, serta mengelola proses penyimpanan dan pengambilan data pada basis data.

Selain itu, Backend Flask bertugas mengatur alur pemrosesan menggunakan Pipeline Scheduler yang dioptimalkan dengan Firefly Algorithm, kemudian mengkoordinasikan proses OCR, Translation, dan Image Text Swapping sesuai dengan layanan yang dipilih pengguna. Backend juga berkomunikasi dengan Gemini API untuk menghasilkan nama, deskripsi, dan kata kunci produk sebelum mengirimkan hasil pemrosesan kembali kepada Chrome Extension maupun Web Dashboard melalui REST API.

Alur kerja implementasi Modul Backend Flask ditunjukkan pada Gambar 3.20, sedangkan spesifikasi modul disajikan pada Tabel 3.6.



Gambar 3.20 Alur Implementasi Modul Chrome Extension

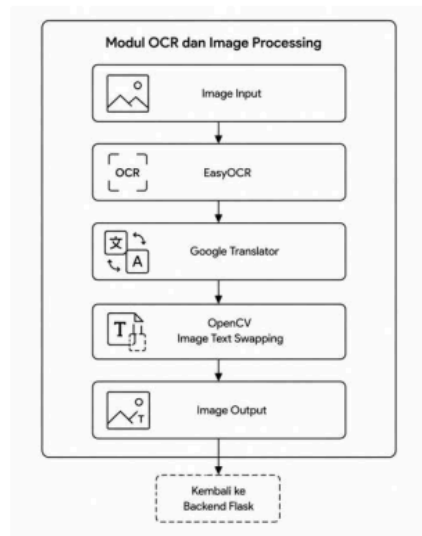
Gambar 3.20 menunjukkan proses dimulai ketika backend menerima permintaan melalui REST API dari Chrome Extension atau Web Dashboard. Selanjutnya backend melakukan validasi request untuk memastikan data yang diterima sesuai dengan kebutuhan sistem. Request kemudian diteruskan ke *Requestdispatcher* untuk menentukan proses yang akan dijalankan berdasarkan endpoint yang diakses. Selanjutnya Pipeline Scheduler yang dioptimalkan menggunakan Firefly Algorithm mengatur urutan pemrosesan OCR, Translation, dan Image Text Swapping. Setelah seluruh proses selesai, data disimpan atau diambil melalui Database Manager, kemudian backend mengembalikan hasil pemrosesan dalam bentuk JSON Response kepada Chrome Extension maupun Web Dashboard melalui REST API.

Tabel 3.6 Spesifikasi Modul Backend Flask

Komponen	Deskripsi
Tujuan	Mengoordinasikan proses pengolahan data, mengelola komunikasi antar modul, serta menyediakan layanan REST API bagi Chrome Extension dan Web Dashboard.
Input	Data hasil DOM scraping dari Chrome Extension dan permintaan dari Web Dashboard.
Proses	REST API → Validasi Request → Request Dispatcher → Pipeline Scheduler (Firefly Algorithm) → OCR → Translation → Image Text Swapping → Database → JSON Response.
Output	Data produk yang telah diproses, hasil OCR, hasil translasi gambar, deskripsi produk, serta respons REST API kepada Chrome Extension dan Web Dashboard.

3.6.3 Modul OCR dan Image Processing

Modul OCR dan Image Processing berfungsi untuk melakukan ekstraksi teks dari gambar produk, menerjemahkan hasil ekstraksi ke bahasa tujuan, serta menghasilkan gambar baru melalui proses Image Text Swapping. Pada penelitian ini, proses ekstraksi teks dirancang menggunakan EasyOCR karena mendukung pengenalan teks multibahasa, termasuk karakter Mandarin yang umum digunakan pada gambar produk platform 1688. Selanjutnya, teks hasil ekstraksi diterjemahkan ke bahasa tujuan menggunakan modul Google Translator sebelum diterapkan kembali ke gambar menggunakan proses Image Text Swapping berbasis OpenCV. Hasil pemrosesan kemudian dikembalikan kepada Backend Flask untuk diteruskan kepada Chrome Extension maupun Web Dashboard.



Gambar 3.21 Perancangan Modul OCR dan Image Processing

Gambar 3.21 menunjukkan alur kerja Modul OCR dan Image Processing. Proses dimulai dengan menerima gambar produk sebagai masukan. Selanjutnya, EasyOCR melakukan ekstraksi teks dari gambar, kemudian hasil ekstraksi diterjemahkan menggunakan Google Translator ke bahasa yang dipilih pengguna. Teks hasil translasi selanjutnya digunakan pada proses Image Text Swapping berbasis OpenCV untuk menghasilkan gambar baru dengan teks yang telah diterjemahkan. Hasil akhir modul berupa gambar produk yang telah diperbarui dan dikembalikan kepada Backend Flask untuk diproses lebih lanjut.

Tabel 3.7 Spesifikasi Modul OCR dan Image Processing

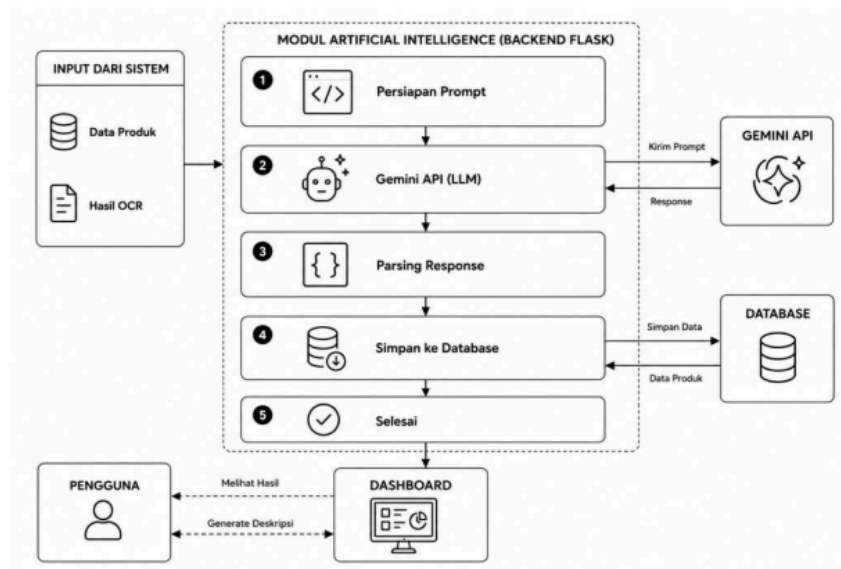
Komponen	Deskripsi
Tujuan	Mengekstraksi teks dari gambar, menerjemahkan teks ke bahasa tujuan, serta menghasilkan gambar hasil <i>Image Text Swapping</i> .
Input	Gambar produk yang diterima dari Backend Flask.
Proses	EasyOCR → Google Translator → OpenCV Image Text Swapping.
Output	Gambar hasil <i>Image Text Swapping</i> , teks hasil OCR, dan teks hasil translasi yang dikembalikan ke Backend Flask.

3.6.4 Modul Artificial Intelligence

Modul Artificial Intelligence digunakan untuk menghasilkan informasi tambahan berupa nama produk, deskripsi produk, dan kata kunci secara otomatis berdasarkan informasi produk hasil DOM scraping dan teks hasil OCR yang telah diterjemahkan. Modul ini memanfaatkan Gemini API sebagai *Large Language Model* (LLM) untuk menghasilkan konten yang lebih informatif dan sesuai dengan konteks produk.

Proses dimulai dengan penyusunan *prompt* berdasarkan data produk dan hasil OCR yang telah diterjemahkan. Selanjutnya, *prompt* dikirimkan ke Gemini API untuk menghasilkan nama produk, deskripsi produk, dan kata kunci. Respons yang diterima kemudian diproses oleh Backend Flask, disimpan ke dalam basis data, dan ditampilkan pada Web Dashboard sebagai informasi tambahan yang dapat membantu pengguna memahami karakteristik produk serta mendukung proses pencarian dan pengelompokan produk.

Alur kerja Modul Artificial Intelligence ditunjukkan pada Gambar 3.22, sedangkan spesifikasi modul disajikan pada Tabel 3.8.



Gambar 3.22 Alur Implementasi Artificial Intelligence

Gambar 3.22 menunjukkan alur implementasi Modul Artificial Intelligence. Proses dimulai ketika Backend Flask menerima informasi produk hasil DOM scraping dan teks hasil OCR yang telah diterjemahkan sebagai masukan untuk penyusunan *prompt*. Selanjutnya, *prompt* dikirimkan ke Gemini API untuk menghasilkan nama produk, deskripsi produk, dan kata kunci. Respons yang diterima kemudian diproses oleh Backend Flask sebelum disimpan ke dalam basis

data. Hasil akhir selanjutnya ditampilkan pada Web Dashboard sehingga dapat digunakan sebagai informasi tambahan dalam penyusunan katalog produk.

Tabel 3.8 Spesifikasi Modul Artificial Intelligence

Komponen	Deskripsi
Tujuan	Menghasilkan nama produk, deskripsi produk, dan kata kunci secara otomatis menggunakan Gemini API.
Input	Informasi produk hasil DOM scraping dan teks hasil OCR yang telah diterjemahkan.
Proses	Penyusunan <i>Prompt</i> → Pengiriman Permintaan ke Gemini API → Parsing Respons → Penyimpanan Hasil ke Basis Data.
Output	Nama produk, deskripsi produk, dan kata kunci yang dihasilkan oleh Gemini API serta disimpan ke dalam basis data.

3.6.5 Modul *Firefly Algorithm*

Modul *Firefly Algorithm* merupakan modul optimasi yang dirancang untuk menentukan konfigurasi *Pipeline Scheduler* yang sesuai dengan kondisi pemrosesan gambar. Pada penelitian ini, *Firefly Algorithm* digunakan untuk mengoptimalkan jumlah *Download Worker*, *OCR Worker*, *Translate Worker*, dan *Image Text Swapping Worker* sehingga setiap tahapan *pipeline* dapat berjalan secara efisien sesuai dengan beban kerja yang sedang diproses.

Modul *Firefly Algorithm* menerima masukan berupa informasi performa *pipeline* yang diperoleh dari *Performance Monitor*. Informasi tersebut terdiri atas panjang antrean (*queue*), jumlah *active task*, nilai *throughput*, dan jumlah *retry* pada setiap tahapan *pipeline*. Data tersebut digunakan sebagai dasar evaluasi untuk menentukan apakah konfigurasi *Pipeline Scheduler* masih sesuai atau perlu dilakukan proses optimasi.

Proses optimasi dirancang untuk dijalankan ketika *Performance Monitor* mendeteksi adanya penurunan performa pada *pipeline*. Kondisi tersebut meliputi terjadinya *bottleneck* akibat panjang antrean yang melebihi ambang batas, penurunan nilai *throughput*, peningkatan jumlah *retry*, atau tingginya penggunaan memori sistem. Dengan mekanisme tersebut, proses optimasi hanya dilakukan ketika diperlukan sehingga penggunaan sumber daya sistem tetap efisien.

Pada penelitian ini, setiap kandidat solusi (*firefly*) merepresentasikan satu konfigurasi *Pipeline Scheduler* yang terdiri atas kombinasi jumlah *worker* pada setiap tahapan *pipeline*. Representasi kandidat solusi dinyatakan sebagai berikut.

$$X = [D, O, T, S] \quad (3.1)$$

dengan:

D = jumlah *Download Worker*

O = jumlah *OCR Worker*

T = jumlah *Translate Worker*

S = jumlah *Image Text Swapping Worker*

Setiap kandidat solusi kemudian dievaluasi menggunakan fungsi *fitness* berdasarkan kondisi aktual *pipeline* yang diperoleh dari *Performance Monitor*. Nilai *fitness* digunakan untuk membandingkan kualitas setiap konfigurasi *Pipeline Scheduler*. *Firefly* dengan nilai *fitness* yang lebih tinggi dianggap memiliki solusi yang lebih baik sehingga akan menarik *firefly* lain untuk bergerak menuju konfigurasi tersebut hingga diperoleh konfigurasi *Pipeline Scheduler* yang optimal.

3.6.5.1 Fungsi Fitness *Firefly Algorithm*

Fungsi *fitness* digunakan untuk mengevaluasi kualitas setiap kandidat konfigurasi *Pipeline Scheduler* yang dihasilkan oleh *Firefly Algorithm*. Tujuan evaluasi ini adalah menentukan konfigurasi *worker* yang paling sesuai dengan kondisi pemrosesan gambar sehingga proses *pipeline* dapat berjalan secara efisien.

Pada penelitian ini, fungsi *fitness* dirancang berdasarkan empat komponen evaluasi, yaitu Throughput, Parameter Match, Resource Efficiency, dan Retry Rate. Keempat komponen tersebut digunakan untuk menilai kemampuan konfigurasi *worker* dalam menjaga keseimbangan antara kecepatan pemrosesan, kesesuaian kapasitas *worker*, efisiensi penggunaan sumber daya, serta tingkat keberhasilan proses. Indikator penilaian setiap komponen ditunjukkan pada Tabel 3.9.

Tabel 3.9 Indikator Penilaian Komponen Fungsi Fitness

Komponen	Indikator Penilaian
Throughput	Jumlah <i>task</i> yang berhasil diproses per satuan waktu.
Parameter Match	Kesesuaian jumlah <i>worker</i> terhadap panjang <i>queue</i> dan jumlah <i>active task</i> .

Komponen	Indikator Penilaian
Resource Efficiency	Tingkat penggunaan <i>worker</i> dan memori sistem selama proses berlangsung.
Retry Rate	Jumlah <i>task</i> yang mengalami <i>retry</i> atau diproses ulang.

Persamaan fungsi *fitness* yang digunakan pada penelitian ini ditunjukkan pada Persamaan (3.1).

$$Fitness = w_1 F_{throughput} + w_2 F_{match} + w_3 F_{resource} + w_4 F_{retry} \quad (3.2)$$

dengan ketentuan

$$w_1 + w_2 + w_3 + w_4 = 1 \quad (3.3)$$

Keterangan:

$F_{throughput}$ = Nilai evaluasi *throughput*.

F_{match} = Nilai kesesuaian konfigurasi worker dengan kebutuhan pemrosesan.

$F_{resource}$ = Nilai efisiensi penggunaan sumber daya sistem.

F_{retry} = Nilai evaluasi berdasarkan tingkat *retry*.

$w_1 + w_2 + w_3 + w_4$ = Bobot masing-masing komponen evaluasi.

Komponen Throughput digunakan untuk mengevaluasi kemampuan sistem dalam menyelesaikan proses pengolahan gambar pada setiap tahapan *pipeline*. Semakin tinggi jumlah tugas yang dapat diproses dalam periode waktu tertentu, semakin tinggi pula nilai evaluasi yang diperoleh.

Komponen Parameter Match digunakan untuk mengevaluasi kesesuaian antara kapasitas *worker* dengan beban kerja pada setiap tahapan *pipeline*. Penilaian dilakukan dengan membandingkan jumlah *worker* yang tersedia terhadap kebutuhan pemrosesan yang ditunjukkan oleh panjang antrean (*queue*) dan jumlah *active task*. Konfigurasi *worker* yang terlalu sedikit dapat menyebabkan *bottleneck*, sedangkan konfigurasi yang terlalu besar mengakibatkan penggunaan sumber daya yang kurang efisien.

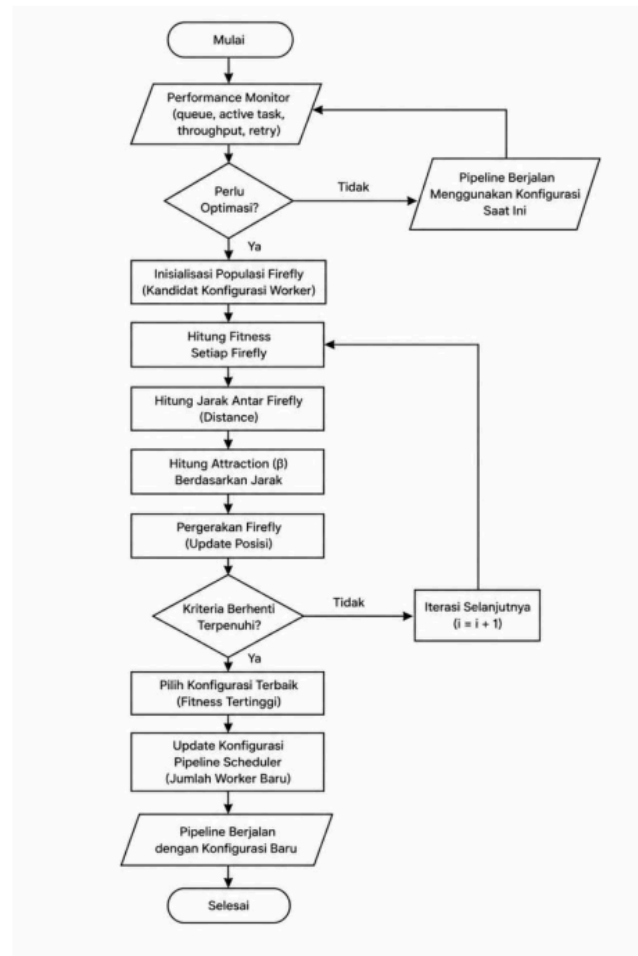
Komponen Resource Efficiency digunakan untuk mengevaluasi efisiensi penggunaan sumber daya selama proses pengolahan gambar berlangsung. Penilaian dilakukan berdasarkan kesesuaian jumlah *worker* terhadap beban kerja pada setiap tahapan *pipeline* serta mempertimbangkan kondisi penggunaan memori sistem agar proses optimasi tidak menghasilkan konfigurasi yang membebani sumber daya secara berlebihan.

Komponen Retry Rate digunakan untuk mengevaluasi tingkat keberhasilan proses berdasarkan jumlah tugas yang harus diproses ulang (*retry*). Semakin sedikit proses *retry* yang terjadi, semakin tinggi nilai evaluasi yang diperoleh sehingga konfigurasi tersebut dinilai lebih stabil.

Berdasarkan keempat komponen tersebut, setiap kandidat solusi memperoleh nilai *fitness* yang berbeda. Konfigurasi dengan nilai *fitness* tertinggi dipilih sebagai konfigurasi terbaik untuk memperbarui jumlah *Download Worker*, *OCR Worker*, *Translate Worker*, dan *Image Text Swapping Worker* pada *Pipeline Scheduler*.

3.6.5.2 Alur Optimasi Firefly Algorithm

Alur optimasi menggunakan Firefly Algorithm ⁶ ditunjukkan pada Gambar 3.23.



Gambar 3.23 Alur Optimasi Firefly Algorithm

Berdasarkan Gambar 3.23, proses optimasi dimulai ketika Performance Monitor melakukan pemantauan terhadap kondisi *Pipeline Scheduler*. Informasi yang dipantau meliputi panjang antrean (*queue*), jumlah *active task*, nilai *throughput*, dan jumlah *retry* pada setiap tahapan *pipeline*. Informasi tersebut digunakan sebagai dasar untuk menentukan apakah konfigurasi *Pipeline Scheduler* masih mampu menangani beban kerja yang sedang diproses.

Apabila kondisi *pipeline* masih berada dalam batas yang telah ditentukan, proses pengolahan gambar akan terus berjalan menggunakan konfigurasi *worker* yang sedang digunakan. Sebaliknya, apabila terdeteksi penurunan performa, sistem akan memulai proses optimasi menggunakan Firefly Algorithm.

Tahap pertama dalam proses optimasi adalah membangkitkan sejumlah kandidat solusi (*firefly*). Setiap *firefly* merepresentasikan kombinasi jumlah *Download Worker*, *OCR Worker*, *Translate Worker*, dan *Image Text Swapping Worker* yang akan digunakan oleh *Pipeline Scheduler*. Selanjutnya setiap kandidat solusi dievaluasi menggunakan fungsi *fitness* untuk mengetahui kualitas konfigurasi yang dihasilkan.

Setelah seluruh kandidat solusi memperoleh nilai *fitness*, Firefly Algorithm melakukan proses pembaruan posisi (*position update*) berdasarkan mekanisme perpindahan antar-*firefly*. Proses ini dilakukan secara berulang sehingga setiap kandidat solusi bergerak menuju konfigurasi yang memiliki nilai *fitness* lebih baik. Pada setiap iterasi, nilai *fitness* dihitung kembali untuk mengevaluasi kualitas konfigurasi terbaru.

Proses optimasi berlangsung hingga memenuhi kriteria penghentian (*stopping criteria*), yaitu jumlah iterasi maksimum telah tercapai atau tidak ditemukan peningkatan nilai *fitness* yang signifikan. Setelah proses optimasi selesai, konfigurasi dengan nilai *fitness* terbaik dipilih sebagai konfigurasi baru *Pipeline Scheduler*.

Konfigurasi tersebut kemudian digunakan untuk memperbarui jumlah *Download Worker*, *OCR Worker*, *Translate Worker*, dan *Image Text Swapping Worker* sehingga proses pengolahan gambar selanjutnya dapat menyesuaikan dengan kondisi beban kerja yang sedang berlangsung. Dengan mekanisme tersebut, sistem diharapkan mampu menjaga kinerja *pipeline* secara adaptif tanpa perlu melakukan konfigurasi secara manual.

Tabel 3.10 Spesifikasi Modul Firefly Algorithm

Komponen	Deskripsi
Tujuan	Mengoptimalkan konfigurasi Pipeline Scheduler menggunakan Firefly Algorithm.
Input	<i>Queue, active task, throughput, retry.</i>
Proses	Deteksi bottleneck → Inisialisasi populasi → Perhitungan fitness → Pergerakan firefly → Seleksi konfigurasi terbaik → Pembaruan Pipeline Scheduler.
Output	Konfigurasi Pipeline Scheduler optimal berupa jumlah worker pada setiap tahapan pipeline.

3.7 Jadwal Penelitian

Penelitian ini direncanakan akan dilaksanakan selama 10 bulan, dimulai dari September 2025 hingga Juni 2026. Adapun jadwal kegiatan penelitian dapat dilihat pada berikut:

Tabel 3.11 Jadwal Penelitian

Tahapan Penelitian	Bulan 1		Bulan 2		Bulan 3		Bulan 4		Bulan 5		Bulan ⁶¹ 6		Bulan 7		Bulan 8		Bulan 9		Bulan 10	
Minggu ke-	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4	1 & 2	3 & 4
Identifikasi Masalah																				
Studi Literatur																				
Analisis Kebutuhan Sistem																				
Perancangan Sistem																				
Implementasi Sistem																				
Pengujian dan Validasi																				
Analisis hasil dan penyusunan laporan akhir																				

BAB 4. HASIL DAN PEMBAHASAN

Bab ini membahas hasil penelitian berupa implementasi dan pengujian sistem yang dikembangkan berdasarkan perancangan pada Bab III. Tahap perancangan dilakukan untuk menghasilkan desain sistem yang menjadi acuan dalam proses pengembangan, sedangkan tahap implementasi menjelaskan realisasi rancangan ke dalam bentuk perangkat lunak. Selanjutnya, pengujian dilakukan untuk memastikan bahwa seluruh fungsi dan komponen sistem telah berjalan sesuai dengan kebutuhan yang telah ditetapkan.

Pembahasan pada bab ini disusun secara bertahap, dimulai dari implementasi sistem hingga pengujian sistem menggunakan pendekatan *V-Model*. Dengan demikian, setiap hasil implementasi dapat divalidasi berdasarkan kebutuhan sistem yang telah dirumuskan pada tahap analisis.

4.1 Implementasi Lingkungan Pengembangan

Implementasi sistem merupakan tahap realisasi dari hasil perancangan yang telah dibahas pada bab sebelumnya. Pada tahap ini, seluruh komponen sistem diimplementasikan menjadi sebuah aplikasi yang mampu mendukung proses pengambilan, pengolahan, dan penyajian informasi produk dari platform 1688.com. Implementasi dilakukan dengan mengintegrasikan setiap modul sesuai arsitektur sistem sehingga seluruh proses dapat berjalan secara otomatis.

Sistem yang dikembangkan terdiri atas beberapa komponen utama, yaitu Chrome Extension sebagai media DOM scraping dan interaksi dengan halaman produk, Backend Flask sebagai pusat pengendali proses dan penyedia layanan REST API, modul OCR dan Image Processing untuk mengekstraksi, menerjemahkan, dan mengganti teks pada gambar produk, modul Gemini API untuk menghasilkan nama produk, deskripsi, dan kata kunci secara otomatis, modul Firefly Algorithm untuk mengoptimalkan konfigurasi *pipeline scheduler* pada proses scraping dan pengolahan data, serta Web Dashboard sebagai media pengelolaan dan penyajian informasi produk.

Seluruh komponen tersebut diintegrasikan melalui REST API sehingga proses pertukaran data antar modul dapat berlangsung secara otomatis, terstruktur, dan konsisten.

4.1.1 Implementasi Lingkungan Pengembangan

Implementasi sistem pada penelitian ini dilakukan menggunakan lingkungan pengembangan yang terdiri atas perangkat keras (*hardware*) dan perangkat lunak (*software*). Lingkungan pengembangan tersebut digunakan untuk mendukung proses pembangunan, implementasi, pengujian, dan integrasi seluruh

komponen sistem, meliputi Chrome Extension, Backend Flask, modul OCR dan Image Processing, Gemini API, Firefly Algorithm, serta Web Dashboard. Spesifikasi perangkat keras dan perangkat lunak yang digunakan pada penelitian ini disajikan pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Perangkat Keras

No	Komponen	Spesifikasi
1	Perangkat Utama	Laptop Acer Aspire 3 Spin 14
2	Prosesor (CPU)	Intel® Core™ i3-1315U (6 Core, hingga 4,5 GHz)
3	Memori (RAM)	8 GB LPDDR5
4	Penyimpanan	SSD 256 GB
5	Pengolah Grafis (GPU)	Intel® UHD Graphics (Integrated)
6	Layar	14 inci Touchscreen, resolusi 1920 × 1200 piksel (Full HD+)
7	Sistem Operasi	Windows 11 64-bit
8	Koneksi Internet	Wi-Fi 6 dengan kecepatan minimal 20 Mbps

Tabel 4.2 Perangkat Lunak

Perangkat Lunak	Versi	Fungsi
Windows 11	24H2	Sistem operasi
Visual Studio Code	Terbaru	Editor kode
Python	3.13.1	Bahasa pemrograman backend
Flask	3.1.3	Framework backend REST API

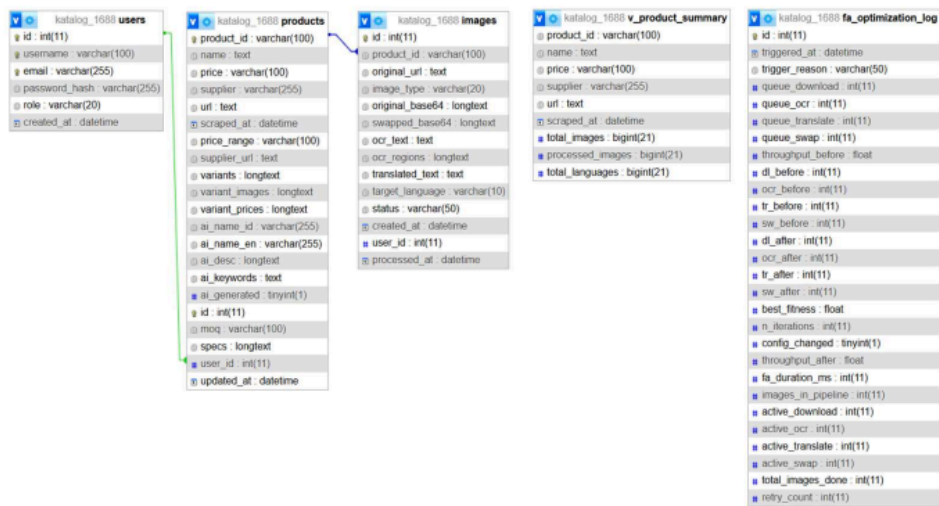
Perangkat Lunak	Versi	Fungsi
Google Chrome	150.0.7871.115	Menjalankan Chrome Extension
EasyOCR	1.7.2	Ekstraksi teks pada gambar
OpenCV	4.13.0	Pengolahan citra digital
Pillow (PIL)	12.2.0	Manipulasi gambar
Google Translate API	v2	Penerjemahan teks hasil OCR
Google Gemini API	gemini-2.5-flash	Generasi deskripsi produk berbasis AI
MySQL	5.2.1	Basis data
XAMPP	3.3.0	Lingkungan pengelolaan basis data MySQL

39

4.2 Implementasi Sistem

4.2.1 Implementasi Basis Data

Basis data¹¹⁵ pada penelitian ini diimplementasikan menggunakan MySQL sebagai media penyimpanan utama untuk mengelola seluruh data yang dihasilkan selama proses scraping, pengolahan gambar, hingga optimasi Pipeline Scheduler. Basis data dirancang menggunakan empat tabel utama, yaitu products, images, users, dan fa_optimization_log, serta satu view yaitu v_product_summary. Relasi⁶⁵ antar tabel dibangun menggunakan primary key dan foreign key untuk menjaga konsistensi data selama proses pengolahan berlangsung.



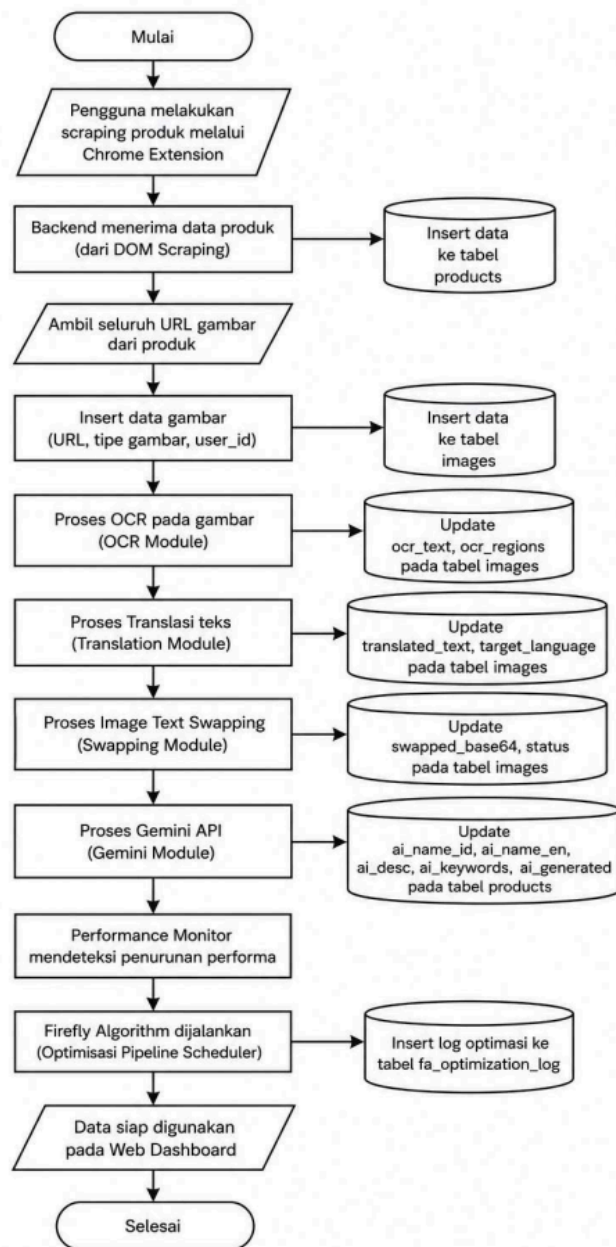
146

Gambar 4.1 Implementasi Basis Data

Berdasarkan Gambar 4.1, tabel products menjadi pusat penyimpanan data karena setiap produk yang berhasil diperoleh dari proses scraping akan direkam terlebih dahulu pada tabel ini. Setiap produk dapat memiliki lebih dari satu gambar sehingga tabel images dihubungkan dengan tabel products melalui atribut product_id. Selain itu, setiap data gambar juga dikaitkan dengan akun pengguna melalui atribut user_id sehingga proses pengolahan gambar dapat ditelusuri berdasarkan pengguna yang menjalankan sistem.

Selama proses pengolahan berlangsung, data pada setiap tabel akan diperbarui secara bertahap sesuai dengan tahapan pipeline. Alur implementasi basis data pada sistem ditunjukkan pada Gambar 4.2.

105



Gambar 4.2 Alur Penyimpanan Data ke Basis Data

Setelah pengguna melakukan proses scraping melalui Chrome Extension, backend menerima data produk berupa nama produk, harga, supplier, URL produk, variasi, serta informasi pendukung lainnya. Data tersebut kemudian disimpan ke dalam tabel products menggunakan product_id sebagai identitas utama setiap produk.

Selanjutnya sistem mengambil seluruh URL gambar dari produk yang berhasil diperoleh. Setiap gambar disimpan sebagai satu record pada tabel images yang dihubungkan dengan tabel products melalui atribut product_id. Pada tahap ini sistem menyimpan URL gambar asli (*original_url*), tipe gambar (*image_type*), serta identitas pengguna yang menjalankan proses scraping.

Setelah data gambar tersimpan, modul OCR memproses setiap gambar untuk mendeteksi teks yang terdapat pada gambar produk. Hasil ekstraksi teks disimpan pada atribut ocr_text, sedangkan koordinat setiap area teks disimpan pada atribut ocr_regions. Data tersebut kemudian digunakan sebagai masukan pada proses translasi.

Tahap berikutnya adalah proses translasi. Hasil translasi setiap teks disimpan pada atribut translated_text, sedangkan bahasa tujuan penyimpanan dicatat pada atribut target_language. Setelah proses translasi selesai, modul Image Text Swapping menghasilkan gambar baru yang telah berisi teks hasil translasi. Gambar hasil pengolahan tersebut kemudian disimpan dalam bentuk Base64 pada atribut swapped_base64, sedangkan atribut status diperbarui sesuai dengan kondisi pemrosesan gambar.

Selain pengolahan gambar, sistem juga mengimplementasikan modul Gemini API untuk menghasilkan nama produk, deskripsi, dan kata kunci secara otomatis berdasarkan data hasil scraping dan OCR. Hasil pemrosesan tersebut digunakan untuk memperbarui atribut ai_name_id, ai_name_en, ai_desc, ai_keywords, serta status ai_generated pada tabel products.

Selama proses pengolahan gambar berlangsung, Performance Monitor terus memantau kondisi Pipeline Scheduler. Ketika Firefly Algorithm dijalankan, sistem akan menyimpan seluruh informasi proses optimasi ke dalam tabel fa_optimization_log. Informasi yang dicatat meliputi kondisi antrean (*queue*) pada setiap tahapan pipeline, nilai throughput, jumlah *active worker*, nilai *fitness* terbaik, jumlah iterasi, durasi optimasi, serta konfigurasi Pipeline Scheduler yang diterapkan setelah proses optimasi selesai. Data tersebut digunakan sebagai dasar analisis pada tahap pengujian untuk membandingkan kondisi pipeline sebelum dan sesudah optimasi.

Untuk meningkatkan efisiensi proses pengambilan data pada Web Dashboard, sistem juga mengimplementasikan view `v_product_summary`. View ini menggabungkan informasi dari tabel `products` dan `images` sehingga dashboard dapat langsung menampilkan jumlah gambar, jumlah gambar yang telah diproses, serta jumlah bahasa hasil translasi tanpa perlu melakukan proses *join* dan agregasi berulang pada setiap permintaan data.

Ringkasan implementasi basis data masing-masing tabel dan *view* ditunjukkan pada Tabel 4.3.

Tabel 4.3 Implementasi Basis Data

Objek Basis Data	Implementasi dalam Sistem
<code>products</code>	Menyimpan data hasil scraping dan diperbarui kembali setelah proses Gemini selesai.
<code>images</code>	Menyimpan data setiap gambar dan diperbarui secara bertahap oleh modul OCR, Translation, dan Image Text Swapping.
<code>users</code>	Digunakan pada proses autentikasi serta mengidentifikasi pemilik data scraping.
<code>fa_optimization_log</code>	Mencatat setiap proses optimasi Firefly Algorithm sebagai data evaluasi performa Pipeline Scheduler.
<code>v_product_summary</code>	Menyediakan data agregat untuk Web Dashboard sehingga proses pengambilan data menjadi lebih efisien.

4.2.2 Implementasi Chrome Extension

Chrome Extension dikembangkan menggunakan HTML, CSS, dan JavaScript dengan mengadopsi arsitektur Manifest V3. Ekstensi berfungsi sebagai antarmuka utama yang menghubungkan pengguna dengan backend sehingga proses scraping, pengiriman data, serta penerimaan hasil pemrosesan dapat dilakukan secara langsung pada halaman produk platform 1688.

Implementasi Chrome Extension dibangun secara modular dengan membagi fungsi ke dalam beberapa berkas sesuai tanggung jawabnya. Struktur direktori Chrome Extension ditunjukkan pada Gambar 4.3.

```
C:\Users\Acer\Documents\1688-assistant\chrome-extension>tree /F
Folder PATH listing
Volume serial number is 00000036 DAC4:F942
C:..
  background.js
  content.js
  manifest.json
  popup.html
  icons
    icon48.png
```

Gambar 4.3 Struktur Direktori Chrome Extension

Berdasarkan Gambar 4.3, Chrome Extension terdiri atas lima komponen utama, yaitu `manifest.json`, `content.js`, `background.js`, `popup.html`, dan folder `icons`. Berkas `manifest.json` digunakan sebagai konfigurasi utama ekstensi, `content.js` bertugas melakukan DOM Scraping dan manipulasi halaman 1688, `background.js` menangani komunikasi dengan Backend Flask, `popup.html` menyediakan antarmuka pengguna, sedangkan folder `icons` berisi ikon yang digunakan sebagai identitas ekstensi pada browser Google Chrome.

Selanjutnya implementasi setiap komponen dijelaskan sebagai berikut.

4.2.2.1 Implementasi Manifest V3

Implementasi Chrome Extension diawali dengan penyusunan berkas `manifest.json` sebagai konfigurasi utama ekstensi. Berkas ini menentukan identitas ekstensi, izin (*permissions*), halaman yang dapat diakses (*host permissions*), *background service worker*, serta *content script* yang akan dijalankan ketika pengguna membuka halaman produk pada platform 1688.

Kode Program 4.1 Konfigurasi Manifest V3

```
{
  "manifest_version": 3,
  "name": "1688 Katalog Assistant",
  "version": "4.0",
  "permissions": ["activeTab", "scripting", "downloads", "storage",
    "alarms"],
  "host_permissions": ["https://*.1688.com/*",
    "http://localhost/*", "http://127.0.0.1/*"],
  "action": { "default_popup": "popup.html", "default_icon": {
    "48": "icons/icon48.png" } },
  "background": { "service_worker": "background.js" },
  "content_scripts": [{
    "matches": ["https://*.1688.com/*"],
    "js": ["content.js"],
    "run_at": "document_idle"
  }]
```

```

  }},
  "icons": { "48": "icons/icon48.png" }
}

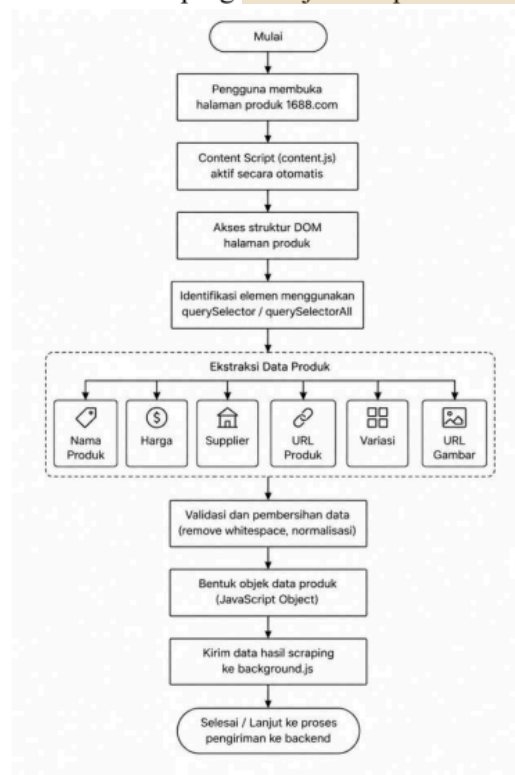
```

Pada implementasi tersebut, properti `content_scripts` digunakan untuk menjalankan `content.js` secara otomatis ketika pengguna membuka halaman produk 1688. Selanjutnya `background.js` dijalankan sebagai *service worker* untuk menangani komunikasi antara Chrome Extension dengan Backend Flask. Sementara itu, *host_permissions* memberikan izin kepada ekstensi untuk mengakses halaman produk pada domain 1688 sehingga proses DOM Scraping dapat dilakukan.

4.2.2.2 Implementasi DOM Scraping

Setelah Chrome Extension aktif pada halaman produk, `content.js` dijalankan secara otomatis untuk mengakses struktur Document Object Model (DOM) yang telah dirender oleh browser. Implementasi DOM Scraping dilakukan menggunakan JavaScript DOM API dengan memanfaatkan fungsi `querySelector()` dan `querySelectorAll()` untuk memperoleh informasi produk dari elemen HTML.

Alur implementasi DOM Scraping ⁶ ditunjukkan pada Gambar 4.4.



Gambar 4.4 Alur implementasi DOM Scraping

Implementasi ekstraksi data dilakukan dengan memilih elemen HTML sesuai struktur halaman produk menggunakan selector CSS. Contoh proses pengambilan elemen ditunjukkan pada Kode Program 4.2.

Kode Program 4.2 Pengambilan Data Menggunakan DOM Scraping

```
function scanShadow(root) {  
  root.querySelectorAll('img').forEach(img => {  
    let src = img.getAttribute('src')  
      || img.getAttribute('data-src')  
      || '';  
    if (src.startsWith('//'))  
      src = 'https:' + src;  
    if (src.includes('cbu01.alicdn.com') ||  
        src.includes('alicdn.com/img/ibank')) {  
      addUrl(src, img);  
    }  
  });  
  root.querySelectorAll('*').forEach(el => {  
    if (el.shadowRoot)  
      scanShadow(el.shadowRoot);  
  });  
}  
document.querySelectorAll('img').forEach(img => {  
  let src = img.getAttribute('src') || '';  
  if (src.startsWith('//'))  
    src = 'https:' + src;  
  if (src.includes('cbu01.alicdn.com') ||  
      src.includes('alicdn.com/img/ibank')) {  
    addUrl(src, img);  
  }  
});
```

Kode Program 4.2 menunjukkan implementasi DOM Scraping yang digunakan untuk mengambil gambar produk dari halaman 1688. Sistem terlebih dahulu mencari seluruh elemen gambar () yang terdapat pada halaman menggunakan fungsi `querySelectorAll()`. Setelah itu, sistem mengambil alamat URL dari setiap gambar melalui atribut `src` atau `data-src`. Apabila URL masih menggunakan format relatif (`//`), sistem akan menambahkan protokol `https:` sehingga URL dapat digunakan pada proses berikutnya.

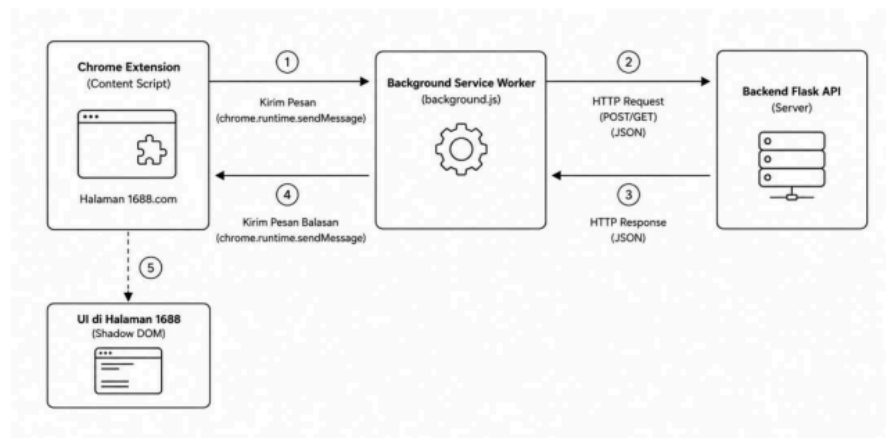
Selanjutnya, sistem melakukan penyaringan terhadap URL gambar yang diperoleh. Hanya gambar yang berasal dari domain penyimpanan gambar produk 1688, yaitu `cbu01.alicdn.com` dan `alicdn.com/img/ibank`, yang akan diproses lebih lanjut. URL yang memenuhi kriteria kemudian disimpan menggunakan fungsi

addUrl() agar tidak terjadi duplikasi data dan siap digunakan pada tahap OCR serta *Image Text Swapping*.

Selain mengambil gambar dari halaman utama, sistem juga melakukan pemindaian pada Shadow DOM melalui fungsi scanShadow(). Hal ini dilakukan karena beberapa gambar produk pada platform 1688 berada di dalam komponen web yang tidak dapat diakses melalui DOM biasa. Dengan melakukan pemindaian secara rekursif pada setiap shadowRoot, sistem dapat memperoleh gambar-gambar tersebut sehingga proses scraping menjadi lebih lengkap dan tidak hanya bergantung pada struktur DOM utama.

4.2.2.3 Implementasi Komunikasi dengan Backend

Komunikasi antara Chrome Extension dan Backend Flask diimplementasikan menggunakan Background Service Worker pada berkas background.js. Komponen ini menerima data hasil DOM Scraping dari content.js, kemudian mengirimkannya ke Backend Flask melalui REST API menggunakan metode HTTP. Alur komunikasi ditunjukkan pada Gambar 4.5.



Gambar 4.5 Alur implementasi DOM Scraping

Berdasarkan Gambar 4.5, proses komunikasi diawali ketika Content Script pada halaman 1688.com mengirimkan data hasil DOM Scraping kepada Background Service Worker melalui mekanisme *message passing* menggunakan `chrome.runtime.sendMessage()` (1). Data yang dikirim berisi informasi produk yang telah berhasil diekstraksi dari halaman web.

Selanjutnya, Background Service Worker meneruskan data tersebut ke Backend Flask API menggunakan *REST API* melalui permintaan HTTP Request dengan format JSON (2). Pada tahap ini, backend menerima data untuk diproses sesuai dengan layanan yang diminta, seperti penyimpanan data, proses OCR, translasi, maupun Image Text Swapping.

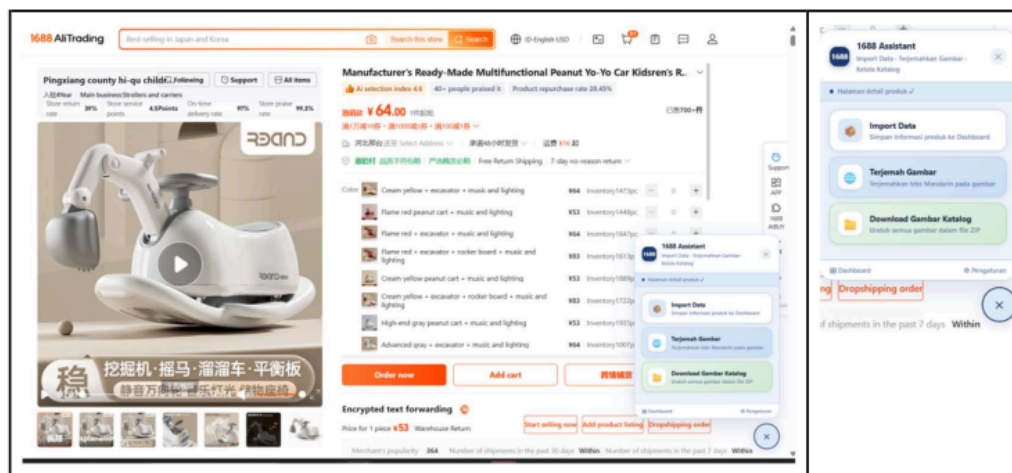
Setelah proses pada backend selesai dilakukan, server mengembalikan hasil pemrosesan dalam bentuk HTTP Response berformat JSON kepada Background Service Worker (3). Respons tersebut kemudian diteruskan kembali ke Content Script menggunakan mekanisme `chrome.runtime.sendMessage()` (4) sehingga hasil pemrosesan dapat diterima oleh ekstensi.

Tahap terakhir, Content Script menampilkan hasil yang diterima ke dalam antarmuka pengguna (UI) yang disisipkan pada halaman 1688.com menggunakan Shadow DOM (5). Dengan mekanisme tersebut, antarmuka ekstensi dapat ditampilkan tanpa mengubah struktur asli halaman web sehingga kompatibilitas dengan halaman 1688.com tetap terjaga.

Melalui mekanisme ini, Background Service Worker berperan sebagai penghubung antara Chrome Extension dan Backend Flask. Pendekatan tersebut membuat proses komunikasi menjadi lebih terstruktur, memisahkan logika antarmuka dengan logika pemrosesan di server, serta memudahkan pengelolaan pertukaran data antara frontend dan backend.

4.2.2.4 Implementasi Antarmuka Chrome Extension

Antarmuka pengguna Chrome Extension diimplementasikan menggunakan `popup.html` yang ditampilkan ketika pengguna memilih ikon ekstensi pada toolbar browser. Popup berfungsi sebagai media interaksi pengguna untuk menjalankan fitur-fitur utama sistem, seperti proses scraping, penerjemahan gambar, akses Web Dashboard, serta konfigurasi sistem.



Gambar 4.6 Implementasi Popup Chrome Extension

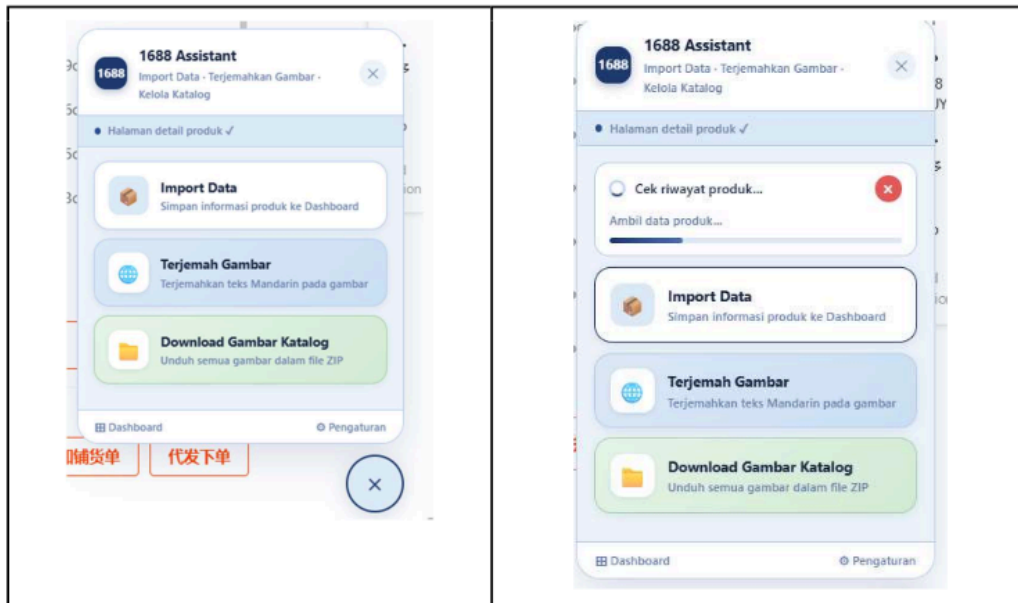
Implementasi antarmuka dibuat sederhana agar seluruh fitur utama dapat diakses tanpa mengganggu aktivitas pengguna pada halaman produk. Setiap tombol

pada popup terhubung dengan fungsi JavaScript yang akan mengirimkan perintah ke content.js atau background.js sesuai proses yang dijalankan.

4.2.2.5 Implementasi Fitur Import Data

Fitur Import Data merupakan salah satu fitur utama pada Chrome Extension yang digunakan untuk mengambil informasi produk dari platform 1688.com dan menyimpannya ke dalam sistem. Implementasi fitur ini tidak hanya mencakup proses pengambilan data dari halaman web, tetapi juga mengintegrasikan beberapa modul sistem, yaitu Content Script, Background Service Worker, Backend Flask, Basis Data, dan Web Dashboard. Integrasi tersebut memungkinkan informasi produk yang diperoleh dari halaman 1688.com dapat diproses, disimpan, dan ditampilkan kembali pada dashboard untuk mendukung pengelolaan katalog produk.

Implementasi antarmuka fitur Import Data pada Chrome Extension ditunjukkan pada Gambar 4.7. Fitur ini disediakan dalam bentuk tombol Import Data yang dapat digunakan pengguna setelah membuka halaman detail produk pada platform 1688.com. Ketika tombol tersebut dipilih, sistem akan menjalankan proses pengambilan informasi produk secara otomatis tanpa memerlukan input manual dari pengguna.

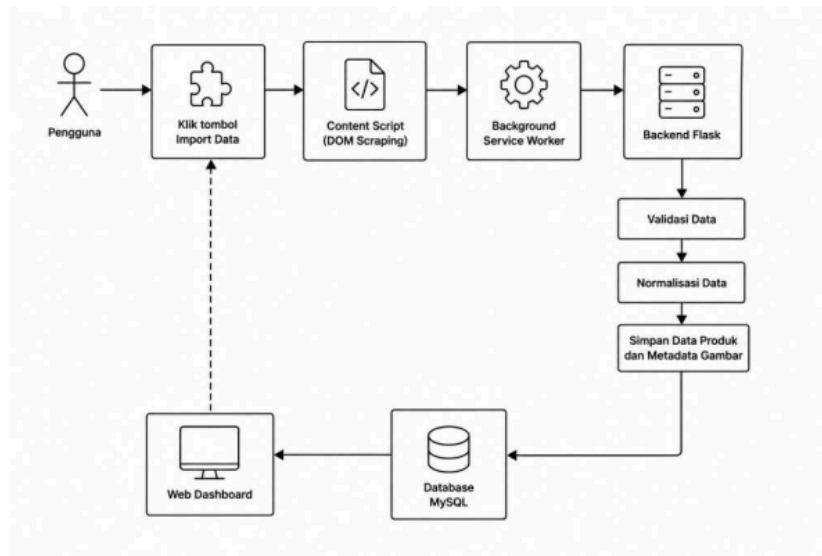


Gambar 4.7 Implementasi Fitur Import Data pada Chrome Extension

Selanjutnya, proses implementasi fitur Import Data melibatkan beberapa modul sistem yang saling berkomunikasi. Proses diawali ketika Content Script melakukan DOM Scraping untuk memperoleh informasi produk, seperti nama

produk, harga, spesifikasi, spp, URL produk, serta URL gambar. Data yang berhasil diperoleh kemudian disusun dalam format JSON sebelum diteruskan ke Background Service Worker melalui mekanisme *message passing*. Selanjutnya, Background Service Worker mengirimkan data tersebut ke Backend Flask menggunakan REST API melalui endpoint `/api/scrape/save`. Pada sisi backend dilakukan proses validasi dan normalisasi data sebelum informasi produk beserta metadata gambar disimpan ke dalam basis data MySQL. Setelah proses penyimpanan selesai, backend mengirimkan respons kepada Chrome Extension sebagai indikator bahwa proses impor data telah berhasil dilakukan.

Agar alur implementasi fitur lebih mudah dipahami, hubungan antar modul pada proses Import Data ditunjukkan pada Gambar 4.8.



Gambar 4.8 Alur Implementasi Fitur Import Data

Berdasarkan Gambar 4.8, implementasi fitur Import Data diawali dengan proses DOM Scraping pada *Content Script* untuk mengambil informasi produk dari halaman 1688.com. Data yang diperoleh kemudian diteruskan ke Background Service Worker dan dikirim ke Backend Flask melalui REST API. Selanjutnya, backend melakukan validasi, normalisasi, serta menyimpan data produk dan metadata gambar ke dalam basis data MySQL sebelum ditampilkan pada Web Dashboard.

Implementasi proses pengambilan informasi produk menggunakan DOM Scraping ditunjukkan pada Kode Program 4.3.

Kode Program 4.3 Implementasi Pengambilan Data Produk

```
function scrapePage() {
```



```

    let name = '';
    ...
    let price = '';
    ...
    let supplier = '';
    ...
    const specs = {};
    return {
      product_name: name,
      price,
      supplier,
      specifications: specs,
      product_url: location.href
    };
  }
}

```

Kode Program 4.3 menunjukkan implementasi proses pengambilan data produk pada halaman detail platform 1688 menggunakan teknik DOM Scraping. Implementasi diawali dengan pencarian elemen HTML yang berisi nama produk menggunakan beberapa selector alternatif untuk menyesuaikan perbedaan struktur halaman. Selanjutnya sistem mengekstraksi informasi harga, nama pemasok (*supplier*), serta spesifikasi produk dari elemen DOM yang tersedia. Setelah seluruh informasi berhasil diperoleh, data disusun ke dalam sebuah objek yang berisi nama produk, harga, pemasok, spesifikasi, dan URL produk sebelum diteruskan ke proses pengiriman data menuju backend.

Selanjutnya, proses pengiriman data hasil scraping menuju Backend Flask diimplementasikan menggunakan Background Service Worker sebagaimana ditunjukkan pada Kode Program 4.4.

Kode Program 4.4 Implementasi Pengiriman Data ke Backend

```

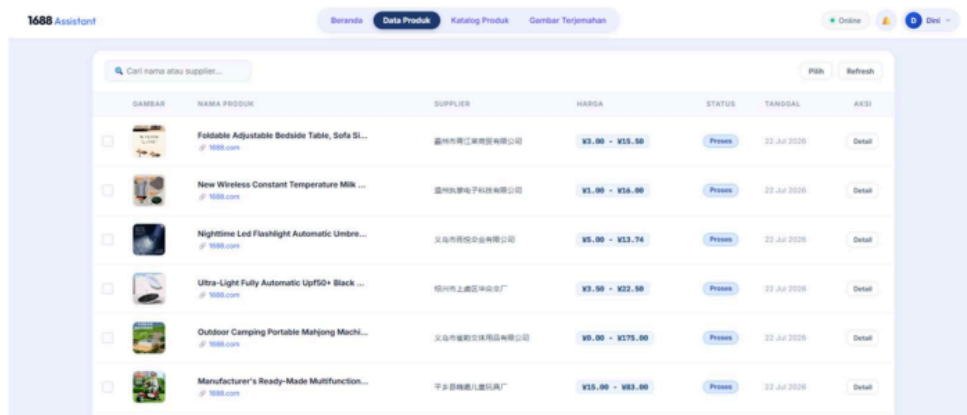
case 'SAVE_SCRAP': {
  const url = cfg.backendUrl;
  try {
    const r = await fetch(`${url}/api/scrape/save`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(msg.data),
      signal: AbortSignal.timeout(15000)
    });
    return sendResponse(await r.json());
  } catch (e) { return sendResponse({ success: false, error:
e.message }); }
}

```

```
}
```

Kode Program 4.4 menunjukkan implementasi proses pengiriman data hasil scraping dari Content Script menuju Backend Flask melalui Background Service Worker. Data produk yang telah diperoleh sebelumnya dikirim menggunakan metode HTTP POST ke endpoint `/api/scrape/save` dalam format JSON melalui fungsi `fetch()`. Pada proses ini sistem menetapkan header `Content-Type: application/json` agar backend dapat mengenali format data yang diterima. Selain itu, mekanisme `AbortSignal.timeout(15000)` diterapkan untuk membatasi waktu tunggu respons selama 15 detik sehingga permintaan yang tidak memperoleh respons dapat dihentikan secara otomatis. Setelah backend selesai memproses permintaan, sistem mengembalikan respons dalam format JSON kepada Chrome Extension sebagai informasi keberhasilan maupun kegagalan proses penyimpanan data.

Setelah seluruh proses selesai, informasi produk yang berhasil disimpan pada basis data akan ditampilkan pada Web Dashboard sebagai pusat pengelolaan katalog produk.



The screenshot shows the '1688 Assistant' web dashboard. At the top, there are tabs for 'Beranda', 'Data Produk', 'Katalog Produk', and 'Gambar Terjemahan'. The 'Data Produk' tab is active. Below the tabs is a search bar with the placeholder 'Cari nama atau supplier...' and buttons for 'Filter' and 'Refresh'. The main content is a table with the following columns: GAMBAR, NAMA PRODUK, SUPPLIER, HARGA, STATUS, TANGGAL, and AKSI. The table contains six rows of product data, each with a small image icon, a product name, a supplier name, a price range, a status (all are 'Proses'), a date (all are '22 Jul 2026'), and an 'Aksi' button labeled 'Detail'.

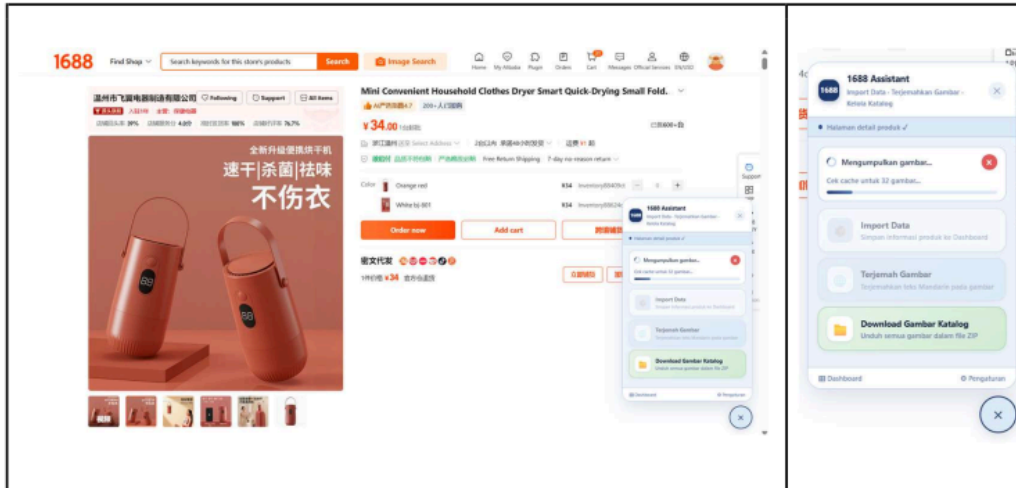
GAMBAR	NAMA PRODUK	SUPPLIER	HARGA	STATUS	TANGGAL	AKSI
	Foldable Adjustable Bedside Table, Sofa St...	温州市隆江家具有限公司	¥3.90 - ¥15.50	Proses	22 Jul 2026	Detail
	New Wireless Constant Temperature Milk ...	温州市隆江家具有限公司	¥1.90 - ¥16.90	Proses	22 Jul 2026	Detail
	Nighttime Led Flashlight Automatic Umbre...	义乌市隆江家具有限公司	¥5.90 - ¥13.74	Proses	22 Jul 2026	Detail
	Ultra-Light Fully Automatic Upt50+ Black ...	温州市隆江家具有限公司	¥3.90 - ¥22.90	Proses	22 Jul 2026	Detail
	Outdoor Camping Portable Mahjong Mech...	义乌市隆江家具有限公司	¥0.90 - ¥175.00	Proses	22 Jul 2026	Detail
	Manufacturer's Ready-Made Multifunction...	平阳县隆江家具有限公司	¥15.00 - ¥81.00	Proses	22 Jul 2026	Detail

Gambar 4.9 Hasil Import Data pada Web Dashboard

4.2.2.6 Implementasi Fitur Terjemahkan Gambar

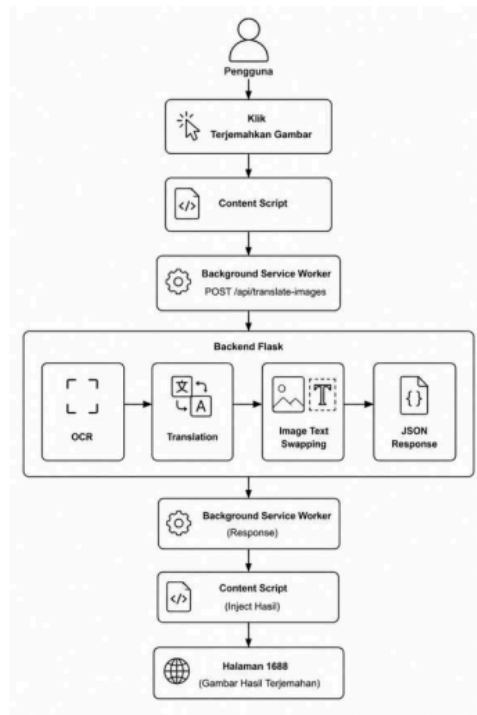
Fitur Terjemahkan Gambar merupakan salah satu fitur utama yang diimplementasikan pada Chrome Extension untuk menerjemahkan teks berbahasa Mandarin yang terdapat pada gambar produk di platform 1688 secara otomatis. Fitur ini membantu pengguna memahami informasi visual produk tanpa perlu melakukan penerjemahan secara manual maupun menggunakan aplikasi penerjemah lain.

Ketika pengguna menekan tombol Terjemahkan Gambar, Chrome Extension terlebih dahulu mengumpulkan URL seluruh gambar produk yang terdapat pada halaman. Selanjutnya, URL gambar tersebut dikirim ke Backend Flask untuk diproses melalui beberapa tahapan, yaitu ekstraksi teks menggunakan Optical Character Recognition (OCR), penerjemahan teks ke bahasa Indonesia, serta penggantian teks pada gambar melalui proses Image Text Swapping. Setelah seluruh proses selesai, gambar hasil terjemahan dikirim kembali ke Chrome Extension dan ditampilkan secara langsung pada halaman produk.



Gambar 4.10 Implementasi Fitur Terjemahkan Gambar

Berdasarkan Gambar 4.10, pengguna dapat menjalankan proses penerjemahan gambar dengan menekan tombol Terjemahkan Gambar pada Chrome Extension. Setelah tombol dipilih, sistem akan mengirimkan URL gambar produk ke backend untuk diproses melalui tahapan OCR, penerjemahan teks, dan Image Text Swapping sebelum hasilnya ditampilkan kembali pada halaman produk.



Gambar 4.11 Alur Implementasi Fitur Terjemahkan Gambar

Berdasarkan Gambar 4.11, implementasi fitur Terjemahkan Gambar diawali ketika pengguna memilih tombol Terjemahkan Gambar pada Chrome Extension. Content Script terlebih dahulu mengumpulkan URL gambar produk yang terdapat pada halaman, kemudian meneruskannya ke Background Service Worker untuk dikirim ke Backend Flask melalui REST API. Selanjutnya backend menjalankan proses OCR untuk mendeteksi teks pada gambar, menerjemahkan hasil OCR ke bahasa Indonesia, serta melakukan proses Image Text Swapping sehingga menghasilkan gambar baru dengan teks hasil terjemahan. Setelah seluruh proses selesai, backend mengirimkan hasilnya dalam format JSON kepada Chrome Extension untuk mengganti gambar asli pada halaman produk secara otomatis.

Kode Program 4.5 Implementasi Pengambilan URL Gambar

```

const quickImgs = [];
const _qSeen = new Set();
document.querySelectorAll('img').forEach(img => {
  let src = img.getAttribute('src') || '';
  if (src.startsWith('//')) src = 'https:' + src;
  const key = src.split('?')[0];
  if ((src.includes('cbu01.alicdn.com') ||
src.includes('alicdn.com/img/ibank'))
    && !_qSeen.has(key)) {
    _qSeen.add(key);
    quickImgs.push({index: quickImgs.length, src, width: 400, height:
  
```

```
400});
    }
  });
});
```

Kode Program 4.5 menunjukkan implementasi proses pengambilan URL gambar produk dari halaman 1688 menggunakan Content Script. Sistem melakukan pemindaian terhadap elemen gambar yang terdapat pada halaman, termasuk gambar yang baru muncul setelah proses *scrolling*. Seluruh URL gambar yang berhasil diperoleh kemudian dikumpulkan ke dalam sebuah daftar dan digunakan sebagai masukan pada proses OCR dan Image Text Swapping di backend.

Kode Program 4.6 Implementasi Pengiriman URL Gambar ke Backend

```
// Background Service Worker - meneruskan request ke backend
case 'TRANSLATE_IMAGES': {
  45 const url = msg.backendUrl || cfg.backendUrl;
  try {
    const r = await fetch(`${url}/api/translate-images`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        images: msg.images,
        target_language: msg.targetLang || 'id',
        product_id: msg.productId || ''
      }),
      signal: AbortSignal.timeout(120000)
    });
    return sendResponse(await r.json());
  } catch(e) { return sendResponse({ success: false, error:
e.message }); }
}
```

Kode Program 4.6 menunjukkan implementasi pengiriman URL gambar dari Chrome Extension menuju Backend Flask menggunakan Background Service Worker. Daftar URL gambar dikirim melalui metode HTTP POST dalam format JSON sehingga backend dapat memproses setiap gambar melalui tahapan OCR, penerjemahan, dan Image Text Swapping. Setelah seluruh proses selesai, backend mengembalikan informasi hasil translasi kepada Chrome Extension untuk ditampilkan kembali pada halaman produk.

Kode Program 4.7 Implementasi Penggantian Gambar

```
async function injectTranslated(results) {
  let replaced = 0;
  for (const item of results) {
    if (!item.translated_image_url && !item.translated_image_base64)
    continue;
  }
}
```

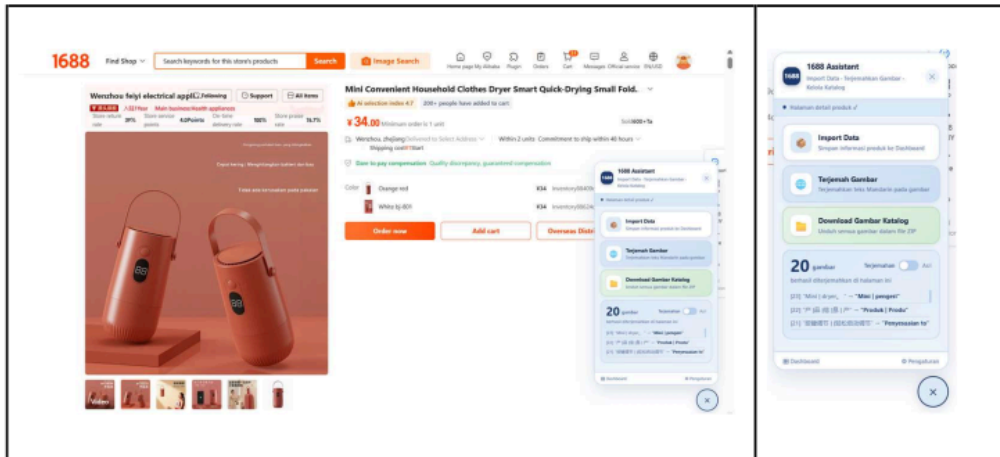


```

const translatedSrc = item.translated_image_url ||
`data:image/jpeg;base64,${item.translated_image_base64}`;
const res = await sendBg({action:'INJECT_IMAGE',
originalSrc:item.original_src,
translatedB64:item.translated_image_base64||item.translated_image_url.replac
e('data:image/jpeg;base64','')});
if (res.success && res.replaced > 0) replaced += res.replaced;
}
return replaced;
}

```

Kode Program 4.7 menunjukkan implementasi proses penggantian gambar asli dengan gambar hasil translasi pada halaman produk. Setelah menerima respons dari backend, Content Script memperbarui atribut gambar pada elemen DOM sehingga gambar yang ditampilkan kepada pengguna merupakan hasil Image Text Swapping tanpa perlu memuat ulang halaman.



Gambar 4.12 Hasil Implementasi Fitur Terjemahkan Gambar

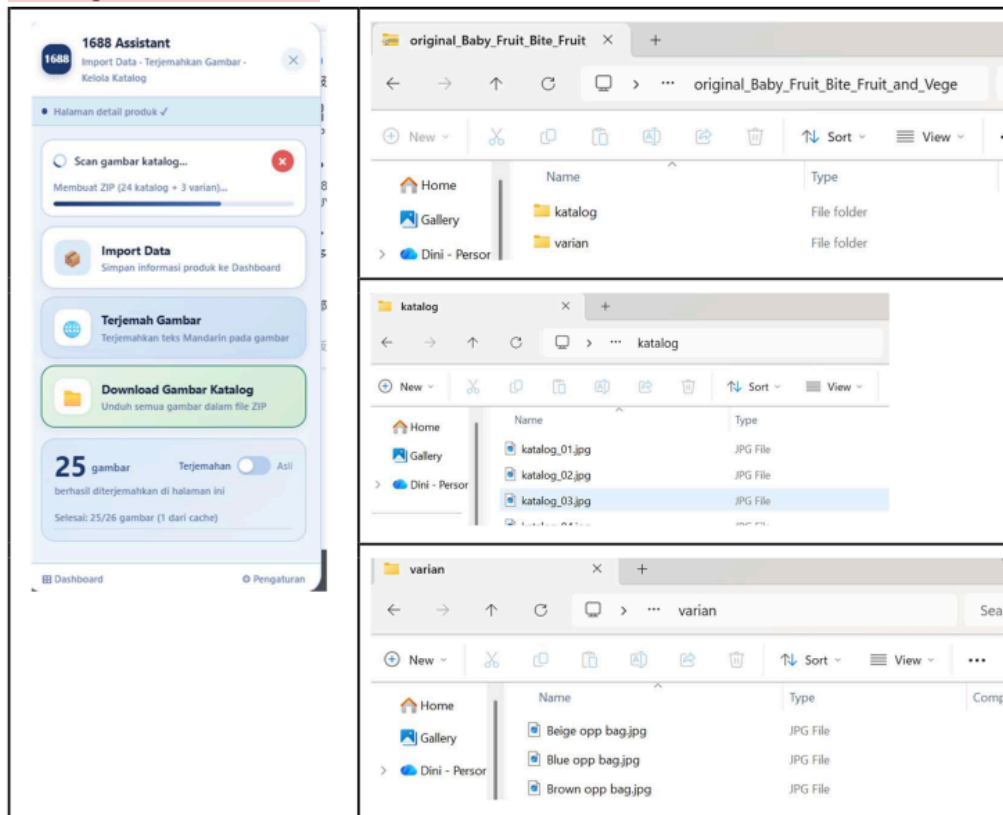
Berdasarkan Gambar 4.12, gambar produk yang semula berisi teks berbahasa Mandarin berhasil ditampilkan kembali dengan teks hasil terjemahan sesuai bahasa yang dipilih pengguna. Hasil translasi ditampilkan secara langsung pada halaman 1688 melalui Chrome Extension tanpa mengubah tata letak halaman maupun mengganggu proses navigasi pengguna.

4.2.2.7 Implementasi Fitur Download Gambar Katalog

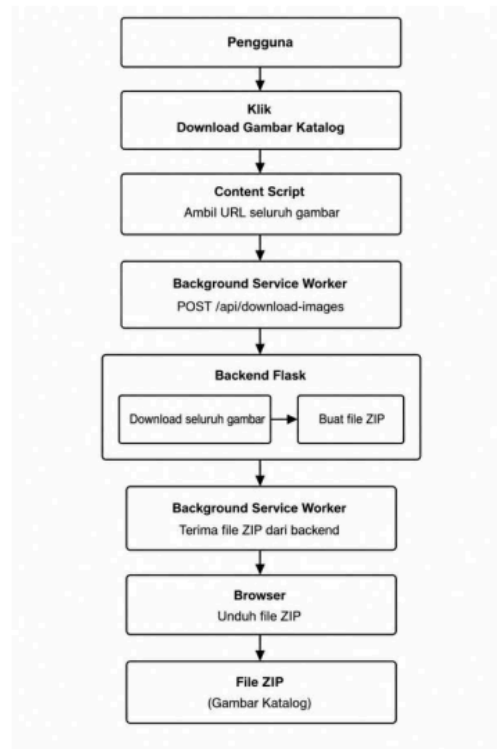
Fitur Download Gambar Katalog diimplementasikan untuk memudahkan pengguna mengunduh seluruh gambar produk yang terdapat pada halaman 1688

secara otomatis. Berbeda dengan proses penyimpanan gambar secara manual satu per satu, fitur ini memungkinkan Chrome Extension mengumpulkan seluruh URL gambar produk dan mengirimkannya ke backend untuk diproses menjadi satu berkas arsip (*ZIP*). Setelah proses selesai, berkas hasil kompresi dikirim kembali kepada pengguna sehingga seluruh gambar katalog dapat diunduh dalam satu kali proses.

Tampilan hasil implementasi fitur Download Gambar Katalog pada Chrome Extension ditunjukkan pada Gambar 4.13, sedangkan alur proses pengunduhan gambar dari Chrome Extension hingga backend menghasilkan berkas ZIP dapat dilihat pada Gambar 4.14.



Gambar 4.13 Implementasi Fitur Download Gambar Katalog



Gambar 4.14 Alur Implementasi Fitur Download Gambar Katalog

Berdasarkan Gambar 4.14, implementasi fitur Download Gambar Katalog diawali ketika pengguna menekan tombol Download Gambar pada Chrome Extension. Selanjutnya Content Script mengumpulkan URL seluruh gambar produk yang terdapat pada halaman dan meneruskannya ke Background Service Worker. ⁵ Data tersebut kemudian dikirim ke Backend Flask melalui REST API untuk diproses. Backend mengunduh seluruh gambar berdasarkan URL yang diterima, kemudian menggabungkannya ke dalam satu berkas berformat ZIP. Setelah proses selesai, berkas ZIP dikirim kembali kepada Chrome Extension dan secara otomatis diunduh oleh browser.

Kode Program 4.8 Implementasi Pengiriman Permintaan Download

```

const total = catalogUrls.length + variantUrls.length;
const isTranslatedMode = !_toggleState && translationMap.size > 0;
const downloadMode = isTranslatedMode ? 'translated' : 'original';

const 16resp = await fetch(`${backendUrl}/api/download-zip-from-urls`, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    catalog: catalogUrls,

```

```

        variant: variantUrls,
        variant_names: variantNameMap,
        name: pageData.name || productId,
        product_id: productId,
        mode: downloadMode
    })
});

```

Kode Program 4.8 menunjukkan implementasi pengiriman permintaan download gambar dari Chrome Extension menuju Backend Flask melalui Background Service Worker. Daftar URL gambar produk dikirim menggunakan metode HTTP POST dalam format JSON menuju endpoint yang menangani proses download. Selanjutnya backend mengunduh seluruh gambar dan mengompresnya menjadi satu berkas ZIP sebelum mengirimkan respons kembali ke Chrome Extension.

Kode Program 4.9 Implementasi Pengunduhan Berkas

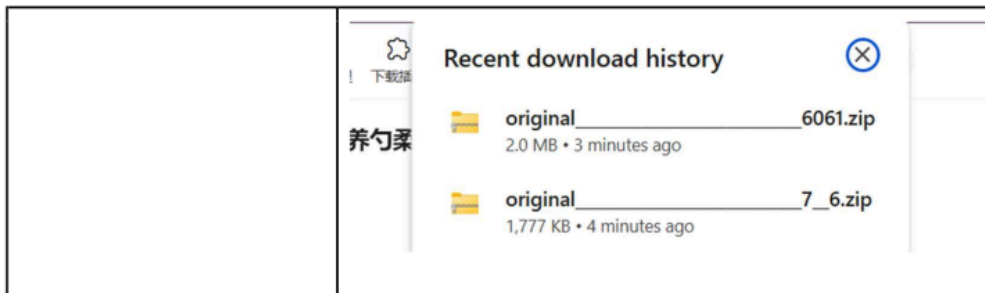
```

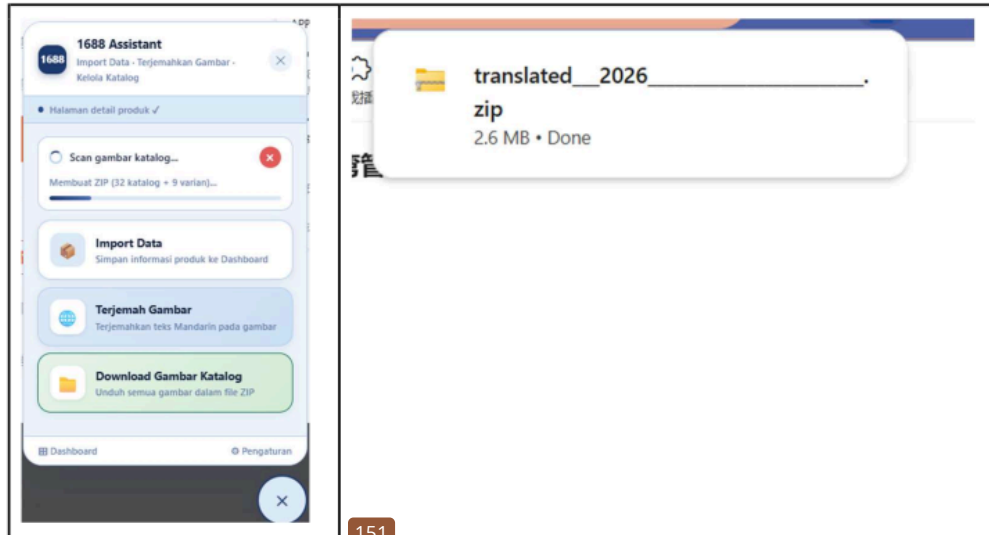
25 if (!resp.ok) throw new Error('Server error');
    const blob = await resp.blob();
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;

    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);

```

Kode Program 4.9 menunjukkan implementasi proses pengunduhan berkas hasil kompresi. Setelah menerima respons dari backend, Chrome Extension membuat tautan unduhan secara otomatis dan menjalankan proses download sehingga pengguna memperoleh seluruh gambar produk dalam satu berkas ZIP tanpa perlu mengunduh gambar satu per satu.





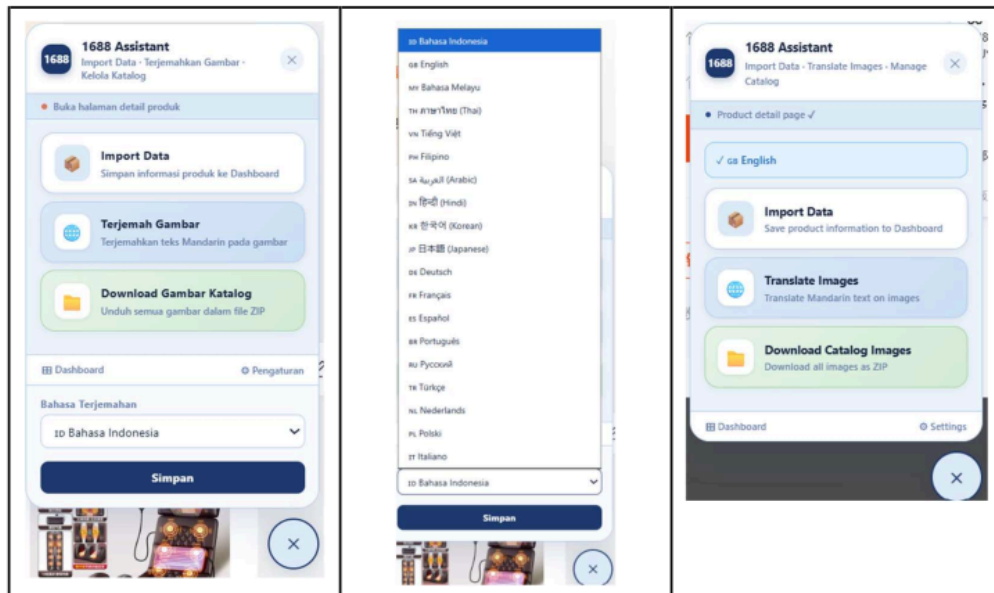
151
Gambar 4.15 Hasil Implementasi

Berdasarkan Gambar 4.15, sistem berhasil mengunduh seluruh gambar katalog produk dan menghasilkan satu berkas ZIP yang berisi seluruh gambar tersebut. Implementasi ini memungkinkan proses pengunduhan dilakukan secara otomatis dalam satu kali operasi sehingga pengguna tidak perlu mengunduh setiap gambar secara terpisah. Dengan demikian, proses pengambilan dan penyimpanan gambar katalog menjadi lebih efisien serta memudahkan pengguna dalam memperoleh seluruh aset gambar produk.

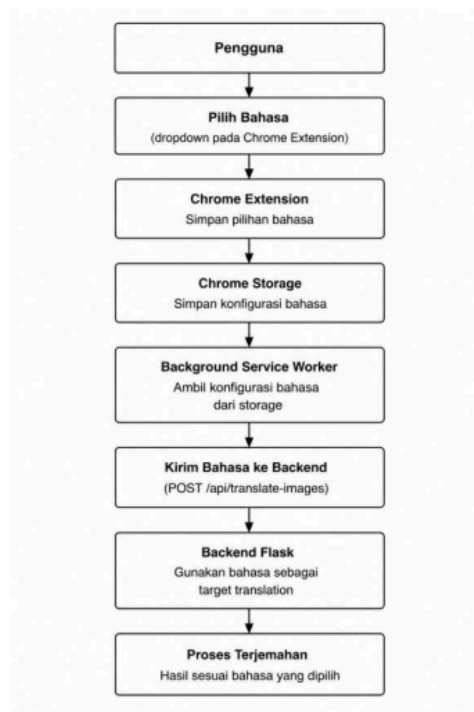
4.2.2.8 Implementasi Pengaturan Bahasa

Fitur Pengaturan Bahasa diimplementasikan untuk memberikan fleksibilitas kepada pengguna dalam menentukan bahasa hasil terjemahan yang akan digunakan pada proses penerjemahan gambar produk. Bahasa yang dipilih akan digunakan sebagai bahasa tujuan pada proses translasi sehingga hasil terjemahan dapat disesuaikan dengan kebutuhan pengguna. Pengaturan ini disimpan oleh Chrome Extension sehingga dapat digunakan kembali pada proses penerjemahan berikutnya tanpa perlu melakukan konfigurasi ulang.

Tabel 4.4 Antarmuka Pengaturan Bahasa



Berdasarkan Tabel 4.4, pengguna dapat memilih bahasa tujuan yang diinginkan melalui antarmuka Chrome Extension. Alur penyimpanan dan penggunaan pengaturan bahasa tersebut dalam proses penerjemahan ditunjukkan pada Gambar 4.16.



Gambar 4.16 Alur Implementasi Pengaturan Bahasa

Berdasarkan Gambar 4.16, implementasi fitur Pengaturan Bahasa diawali ketika pengguna memilih bahasa tujuan melalui antarmuka Chrome Extension. Selanjutnya pilihan bahasa disimpan menggunakan Chrome Storage sehingga tetap tersedia meskipun halaman dimuat ulang. Ketika pengguna menjalankan proses Terjemahkan Gambar, Background Service Worker mengambil konfigurasi bahasa yang tersimpan dan mengirimkannya ke Backend Flask sebagai parameter bahasa tujuan. Backend kemudian menggunakan parameter tersebut sebagai acuan pada proses penerjemahan teks.

Kode Program 4.10 Implementasi Penyimpanan Pengaturan Bahasa

```
// File: background.js
// Menyimpan konfigurasi bahasa ke Chrome Storage
case 'SAVE_CONFIG':
  await chrome.storage.local.set({
    backendUrl: msg.backendUrl || cfg.backendUrl,
    targetLang: msg.targetLang || cfg.targetLang
  });

  return sendResponse({
    success: true
  });
```

Kode Program 4.10 menunjukkan implementasi penyimpanan pengaturan bahasa pada file background.js. Ketika pengguna memilih bahasa tujuan melalui antarmuka Chrome Extension, nilai bahasa disimpan ke dalam Chrome Storage menggunakan `chrome.storage.local.set()`. Dengan mekanisme tersebut, konfigurasi bahasa tetap tersedia pada penggunaan berikutnya tanpa perlu dilakukan pengaturan ulang.

Kode Program 4.11 Implementasi Pengiriman Bahasa ke Backend

```
// File: background.js
// Mengambil konfigurasi bahasa yang tersimpan
const cfg = await getConfig();
// Mengirim bahasa tujuan ke Backend Flask
const response = await fetch(`${url}/api/translate-images`, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    images: msg.images,
    target_language: msg.targetLang || cfg.targetLang,
    product_id: msg.productId || ''
  })
})
```

```
});
return sendResponse(await response.json());
```

Kode Program 4.11 menunjukkan implementasi pengiriman konfigurasi bahasa pada file background.js. Background Service Worker mengambil nilai bahasa yang telah tersimpan melalui fungsi getConfig(), kemudian mengirimkannya sebagai parameter target_language pada permintaan menuju Backend Flask. Parameter tersebut digunakan sebagai acuan dalam proses penerjemahan teks pada gambar.

Tabel 4.5 Hasil Implementasi Pengaturan Bahasa

Gambar original	Hasil terjemahan Bahasa Indonesia	Hasil terjemahan Bahasa Inggris
		
		

Berdasarkan Tabel 4.5, fitur Pengaturan Bahasa berhasil menerjemahkan ke beberapa pilihan bahasa pengguna dan dengan implementasi tersebut, hasil terjemahan dapat disesuaikan dengan bahasa tujuan yang dipilih pengguna.

4.2.3 Implementasi Modul Sistem

Modul sistem merupakan komponen inti yang bertanggung jawab dalam memproses setiap permintaan yang diterima dari Chrome Extension. Modul ini mengelola berbagai proses, seperti pengolahan data hasil scraping, penerjemahan gambar, ringkasan produk, hingga optimasi proses menggunakan Firefly Algorithm.

Implementasi modul sistem pada penelitian ini menerapkan arsitektur yang bersifat modular sehingga setiap modul memiliki tanggung jawab yang berbeda namun saling terintegrasi dalam satu alur pemrosesan. Pendekatan tersebut mempermudah proses pengembangan, pemeliharaan, serta memungkinkan setiap modul dikembangkan secara independen tanpa memengaruhi modul lainnya.

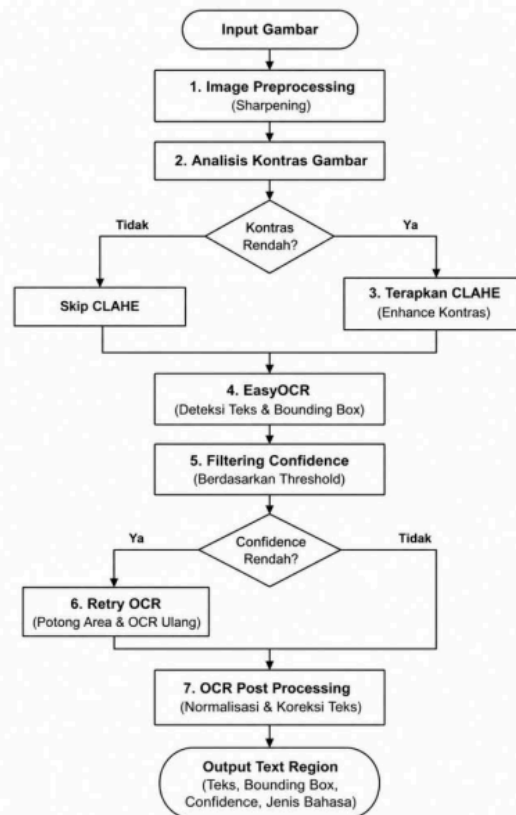
Komunikasi antar modul dilakukan melalui pemanggilan fungsi pada backend, sedangkan pertukaran data dengan Chrome Extension menggunakan layanan REST API dalam format JSON.

Implementasi modul sistem pada penelitian ini meliputi modul OCR dan Image Processing, modul Google Translate API, modul Google Gemini API, modul Firefly Algorithm, serta modul REST API dan pengelolaan basis data. Implementasi masing-masing modul dijelaskan pada subbab berikut.

4.2.3.1 Implementasi Modul OCR

Modul *Optical Character Recognition* (OCR) diimplementasikan untuk mendeteksi serta mengekstraksi teks yang terdapat pada gambar produk dari platform 1688.com. Implementasi modul ini menggunakan library EasyOCR dengan konfigurasi bahasa Mandarin Simplified (ch_sim) dan bahasa Inggris (en) sehingga mampu mengenali karakter yang umum ditemukan pada katalog produk. Sebelum proses OCR dilakukan, sistem terlebih dahulu menerapkan beberapa tahapan prapemrosesan citra (*image preprocessing*) untuk meningkatkan kualitas gambar sehingga proses pengenalan karakter dapat dilakukan dengan lebih optimal. Selain itu, modul OCR pada penelitian ini juga mengimplementasikan mekanisme Retry OCR, OCR Post Processing, serta penggunaan parameter hasil optimasi Firefly Algorithm untuk meningkatkan efisiensi proses ekstraksi teks.

Tahapan implementasi modul OCR ⁶ ditunjukkan pada Gambar 4.17.



Gambar 4.17 Implementasi Modul OCR

Berdasarkan Gambar 4.17, proses OCR diawali dengan menerima masukan berupa gambar produk yang diperoleh dari proses pengunduhan gambar pada platform 1688.com. Gambar tersebut selanjutnya memasuki tahap Image Preprocessing untuk meningkatkan kualitas citra sebelum dilakukan proses pendeteksian teks. Pada implementasi ini, sistem menerapkan proses image sharpening untuk mempertajam detail citra sehingga karakter teks lebih mudah dikenali oleh modul OCR.

Setelah proses *preprocessing* selesai, sistem melakukan Analisis Kontras terhadap gambar. Tahap ini bertujuan menentukan apakah gambar memerlukan peningkatan kontras menggunakan metode Contrast Limited Adaptive Histogram Equalization (CLAHE). Apabila gambar telah memiliki tingkat kontras yang baik, proses CLAHE dilewati sehingga waktu pemrosesan dapat dikurangi. Sebaliknya, apabila kontras gambar rendah, sistem akan menerapkan CLAHE sebelum proses OCR dilakukan. Pendekatan ini memungkinkan proses peningkatan kualitas citra dilakukan secara adaptif sesuai karakteristik gambar yang diproses.

Tahap berikutnya adalah proses EasyOCR, yaitu proses pendeteksian posisi teks (*bounding box*) sekaligus pengenalan karakter pada gambar. Pada implementasinya, EasyOCR menghasilkan informasi berupa posisi teks, isi teks,

serta nilai *confidence* untuk setiap hasil deteksi. Selanjutnya sistem melakukan Filtering Confidence untuk menyaring hasil OCR berdasarkan nilai *confidence threshold* yang telah ditentukan sehingga hanya hasil deteksi yang memenuhi kriteria yang digunakan pada proses berikutnya.

Apabila sistem menemukan region teks dengan nilai *confidence* rendah, maka dijalankan mekanisme Retry OCR. Pada tahap ini sistem melakukan pemotongan ulang (*cropping*) pada area teks berdasarkan koordinat *bounding box* kemudian menjalankan proses OCR kembali pada area tersebut. Hasil OCR ulang akan digunakan apabila memiliki nilai *confidence* yang lebih tinggi dibandingkan hasil deteksi sebelumnya. Pendekatan ini diterapkan untuk meningkatkan akurasi pengenalan teks, khususnya pada teks Mandarin yang berukuran kecil atau memiliki kualitas gambar yang kurang baik.

Tahap terakhir adalah OCR Post Processing, yaitu proses normalisasi hasil OCR untuk memperbaiki karakter yang tidak sesuai, menghilangkan karakter yang tidak diperlukan, serta menghasilkan keluaran yang lebih konsisten sebelum diteruskan ke modul translasi. Hasil akhir modul OCR berupa kumpulan informasi yang terdiri atas isi teks, koordinat *bounding box*, nilai *confidence*, serta jenis bahasa pada setiap region teks. Informasi tersebut selanjutnya digunakan oleh modul Google Translate API dan modul Image Text Swapping sebagai dasar proses penerjemahan gambar.

Kode Program 4.12 Implementasi Image Preprocessing

```
class ImagePreprocessor:

    @staticmethod
    def preprocess(image: np.ndarray) -> np.ndarray:
        kernel_sharpen = np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])
        return cv2.filter2D(image, -1, kernel_sharpen)

    @staticmethod
    def enhance_for_ocr(image: np.ndarray) -> np.ndarray:
        lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
        l, a, b = cv2.split(lab)
        clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
        enhanced_lab = cv2.merge((clahe.apply(l), a, b))
        return cv2.cvtColor(enhanced_lab, cv2.COLOR_LAB2BGR)
```

Kode Program 4.12 menunjukkan implementasi tahap *Image Preprocessing*. Sistem terlebih dahulu menerapkan metode *image sharpening* menggunakan fungsi `filter2D()` untuk mempertajam detail citra sehingga karakter teks lebih mudah dikenali. Selanjutnya fungsi `enhance_for_ocr()` menerapkan

metode *Contrast Limited Adaptive Histogram Equalization (CLAHE)* pada kanal luminansi *citra untuk* meningkatkan kontras *gambar* yang memiliki kualitas rendah sebelum diproses oleh modul OCR.

Kode Program 4.13 Implementasi EasyOCR

```
results = self.reader.readtext(
    enhanced,
    detail=1,
    paragraph=False,
    contrast_ths=0.05,
    adjust_contrast=0.7,
    text_threshold=0.6,
    low_text=0.30,
    link_threshold=0.30,
    mag_ratio=self.fa_mag_ratio,
    canvas_size=self.fa_canvas_size,
    size=5,
    add_margin=0.2,
    width_ths=0.3,
    height_ths=0.3,
)
```

Kode Program 4.13 menunjukkan implementasi proses ekstraksi teks menggunakan library EasyOCR. Sistem memproses gambar hasil *preprocessing* untuk mendeteksi posisi teks (*bounding box*), mengenali karakter, serta menghasilkan nilai *confidence*. Implementasi menggunakan parameter seperti *text_threshold*, *low_text*, *link_threshold*, *mag_ratio*, dan *canvas_size* untuk menyesuaikan proses deteksi terhadap karakteristik gambar produk. Nilai *mag_ratio* dan *canvas_size* diperoleh dari hasil optimasi Firefly Algorithm sehingga konfigurasi OCR menjadi lebih optimal.

Kode Program 4.14 Implementasi Retry OCR

```
need_retry = (
    region.is_chinese
    and len(region.text) <= 3
    and region.confidence < 0.30
)

if need_retry and retry_count < MAX_RETRY:
    retry_count += 1
    t = time.perf_counter()
    retry = self._retry_small_region(image, region)
    logger.info(
        f"[PERF] Retry OCR : {time.perf_counter()-
t:.2f}s"
    )
```

```

if retry.confidence > region.confidence:
    region = retry

```

Kode Program 4.14 menunjukkan implementasi mekanisme *Retry OCR*. Sistem terlebih dahulu mengevaluasi hasil OCR berdasarkan panjang teks dan nilai *confidence*. Apabila region teks Mandarin memiliki *confidence* rendah, sistem menjalankan fungsi `_retry_small_region()` untuk melakukan proses *cropping* pada area *bounding box* dan menjalankan EasyOCR kembali pada area tersebut. Hasil OCR ulang digunakan apabila menghasilkan nilai *confidence* yang lebih tinggi dibandingkan hasil sebelumnya sehingga akurasi pengenalan karakter dapat ditingkatkan tanpa mengulang proses OCR terhadap keseluruhan gambar.

Tabel 4.6 Hasil Deteksi Bounding Box Menggunakan EasyOCR

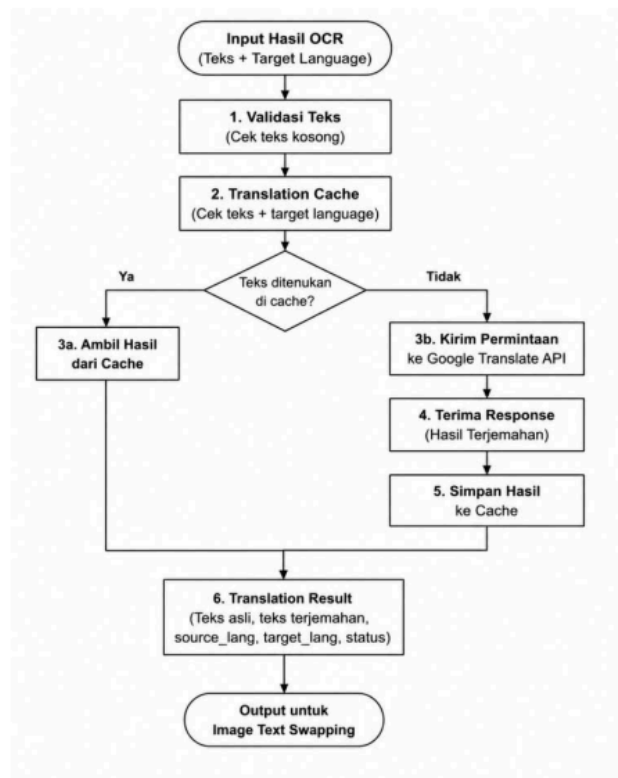
 活力橙 柠檬黄 馥郁红 墨绿色 巴黎蓝 梦幻黑	INFO:pipeline.ocr_extractor:Ditemukan 6 region teks (6 Mandarin, 0 non-Mandarin) Jumlah teks terdeteksi : 6 Region 1 Teks : 活力橙 Bahasa : Mandarin BoundingBox : (62, 279) -> (170, 331) Region 2 Teks : 柠檬黄 Bahasa : Mandarin BoundingBox : (319, 279) -> (427, 330) Region 3 Teks : 馥郁红 Bahasa : Mandarin BoundingBox : (560, 279) -> (670, 330) Region 4 Teks : 墨绿色 Bahasa : Mandarin BoundingBox : (63, 609) -> (168, 659) Region 5 Teks : 巴黎蓝 Bahasa : Mandarin BoundingBox : (323, 609) -> (427, 659) Region 6 Teks : 梦幻黑 Bahasa : Mandarin BoundingBox : (560, 608) -> (667, 659)
--	---

Berdasarkan hasil implementasi, modul OCR mampu mendeteksi posisi teks (*bounding box*) sekaligus mengekstraksi karakter Mandarin yang terdapat pada gambar produk. Informasi yang dihasilkan meliputi isi teks, koordinat *bounding box*, nilai *confidence*, serta jenis bahasa pada setiap region teks. Hasil tersebut selanjutnya digunakan sebagai masukan pada modul Google Translate API untuk proses penerjemahan dan modul Image Text Swapping untuk proses penggantian teks pada gambar.

4.2.3.2 Implementasi Modul Translasi

Modul Translasi diimplementasikan untuk menerjemahkan teks hasil ekstraksi dari modul OCR ke dalam bahasa tujuan yang dipilih oleh pengguna melalui fitur Pengaturan Bahasa pada Chrome Extension. Implementasi modul ini menggunakan layanan Google Translate API yang diakses melalui protokol HTTP menggunakan library Requests pada Python. Modul translasi menerima masukan berupa teks hasil OCR beserta konfigurasi bahasa tujuan, kemudian menghasilkan teks terjemahan yang selanjutnya digunakan oleh modul Image Text Swapping untuk menggantikan teks pada gambar produk.

Untuk meningkatkan efisiensi proses translasi, implementasi modul ini menerapkan mekanisme translation cache, connection pooling, batch translation, dan multithreading sehingga proses penerjemahan beberapa region teks dapat dilakukan secara lebih cepat tanpa melakukan permintaan yang berulang terhadap teks yang sama. Tahapan implementasi modul translasi ⁶ ditunjukkan pada Gambar 4.18.



Gambar 4.18 Implementasi Modul Translasi

Berdasarkan Gambar 4.18, proses translasi diawali dengan menerima masukan berupa hasil ekstraksi teks dari modul OCR. Sistem terlebih dahulu melakukan Validasi Teks untuk memastikan bahwa teks yang diterima tidak kosong sebelum proses penerjemahan dilakukan. Apabila teks tidak memenuhi kriteria, sistem akan mengembalikan teks asli sebagai hasil keluaran.

Selanjutnya sistem memeriksa Translation Cache untuk mengetahui apakah teks yang sama pernah diterjemahkan sebelumnya. Apabila hasil terjemahan telah tersedia (*cache hit*), sistem langsung menggunakan hasil tersebut tanpa mengirimkan permintaan ke Google Translate API. Sebaliknya, apabila hasil translasi belum tersedia (*cache miss*), sistem mengirimkan permintaan ke Google Translate API menggunakan parameter bahasa sumber dan bahasa tujuan yang telah ditentukan.

Setelah proses translasi selesai, hasil terjemahan disimpan kembali ke dalam cache sehingga dapat digunakan pada proses berikutnya apabila ditemukan teks yang sama. Tahap terakhir menghasilkan objek Translation Result yang berisi teks asli, hasil terjemahan, bahasa sumber, bahasa tujuan, serta status proses translasi. Informasi tersebut selanjutnya diteruskan ke modul Image Text Swapping untuk menggantikan teks pada gambar produk.

Kode Program 4.15 Implementasi Google Translate API

```
def translate(self, text, target_lang="id", source_lang="zh-CN"):

    # Validasi teks
    137 if not text or not text.strip():
        return TranslationResult(...)

    # Pemeriksaan translation cache
    cache_key = (text, target_lang)
    68 with self.cache_lock:
        if cache_key in self.cache:
            return self.cache[cache_key]

    # Parameter Google Translate API
    154 params = {"client": "gtx", "sl": source_lang, "tl": target_lang, "dt": "t",
    "q": text}

    # Mengirim permintaan ke Google Translate API
    94 response = self.session.get(self.BASE_URL, params=params, timeout=3)

    translated = "".join(segment[0] for segment in response.json()[0] if
    segment[0])

    result = TranslationResult(
        75 original_text=text, translated_text=translated,
        source_lang=source_lang, target_lang=target_lang,
        method="google", success=True
    )

    # Menyimpan hasil translasi ke cache
    134 self.cache[cache_key] = result

    return result
```

Kode Program 4.15 menunjukkan implementasi proses translasi menggunakan Google Translate API pada file `translator.py`. Sistem terlebih dahulu melakukan validasi terhadap teks masukan kemudian memeriksa translation cache untuk mengetahui apakah hasil terjemahan telah tersedia. Apabila hasil translasi belum ditemukan pada cache, sistem membentuk parameter permintaan yang terdiri atas bahasa sumber (sl), bahasa tujuan (tl), serta teks yang akan diterjemahkan (q). Selanjutnya sistem mengirimkan permintaan HTTP menggunakan metode GET melalui objek Session sehingga koneksi dapat digunakan kembali (*connection pooling*) pada proses translasi berikutnya. Setelah menerima respons dari Google Translate API, sistem mengekstraksi hasil terjemahan dari data JSON kemudian membentuk objek TranslationResult yang berisi teks asli, hasil terjemahan, bahasa sumber, bahasa tujuan, metode translasi, serta status keberhasilan proses. Hasil translasi tersebut kemudian disimpan ke dalam translation cache agar dapat digunakan kembali apabila ditemukan teks yang sama pada proses berikutnya. Selain itu, sistem juga menerapkan mekanisme *fallback*, yaitu mengembalikan teks asli apabila terjadi kesalahan komunikasi dengan layanan Google Translate API sehingga proses selanjutnya tetap dapat dijalankan.

Kode Program 4.16 Implementasi Batch Translation

```
def translate_batch(self, texts, target_lang="id", source_lang="zh-CN"):

    # Menghilangkan teks yang sama
    unique_texts = list(dict.fromkeys(texts))

    def worker(text):
        return self.translate(text=text, target_lang=target_lang,
                               source_lang=source_lang,
                               session=self.session)

    # Menjalankan translasi secara paralel
    with ThreadPoolExecutor(max_workers=min(4, len(unique_texts))) as
    executor:
        results = list(executor.map(worker, unique_texts))

    # Memetakan hasil translasi
    translation_map = {r.original_text: r for r in results}

    # Mengembalikan hasil sesuai urutan OCR
    return [translation_map[text] for text in texts]
```

Kode Program 4.16 menunjukkan implementasi proses Batch Translation. Sebelum proses translasi dilakukan, sistem terlebih dahulu menghilangkan teks yang memiliki nilai sama sehingga hanya teks unik yang dikirimkan ke Google Translate API. Pendekatan ini bertujuan mengurangi jumlah permintaan ke layanan translasi dan meningkatkan efisiensi proses. Selanjutnya sistem menggunakan `ThreadPoolExecutor` untuk menjalankan beberapa proses translasi secara paralel (*multithreading*) sehingga waktu pemrosesan dapat dipersingkat ketika gambar mengandung banyak region teks. Setelah seluruh proses translasi selesai, hasil terjemahan dipetakan kembali sesuai urutan hasil OCR sehingga hubungan antara teks, posisi *bounding box*, dan hasil terjemahan tetap terjaga sebelum diteruskan ke modul Image Text Swapping.

Selanjutnya sistem menggunakan `ThreadPoolExecutor` untuk menjalankan beberapa proses translasi secara paralel sehingga waktu pemrosesan dapat dipersingkat ketika gambar mengandung banyak region teks. Setelah seluruh proses translasi selesai, hasil terjemahan dipetakan kembali sesuai urutan hasil OCR sehingga hubungan antara teks, posisi *bounding box*, dan hasil terjemahan tetap terjaga sebelum diteruskan ke modul Image Text Swapping.

Tabel 4.7 Hasil Implementasi Modul Translasi

	Ditemukan 7 region teks (7 Mandarin, 0 non-Mandarin) Ditemukan 7 teks GOOGLE TRANSLATE OCR : 低功耗强续航 Translated : Konsumsi daya rendah dan masa pakai baterai yang lama GOOGLE TRANSLATE OCR : 7天畅快用 Translated : Nikmati selama 7 hari GOOGLE TRANSLATE OCR : 蓝牙低功耗芯片。耗能低, Translated : Chip Bluetooth hemat energi. Konsumsi energi rendah, GOOGLE TRANSLATE OCR : 性能稳定; Translated : Kinerja yang stabil; GOOGLE TRANSLATE OCR : 一次充电 Translated : Satu biaya GOOGLE TRANSLATE OCR : 可连续听歌12小时7天长待机 , Translated : Dapat mendengarkan musik terus menerus selama 12 jam 7 hari dengan waktu standby yang lama. GOOGLE TRANSLATE OCR : 满足长途出行 Translated : Cocok untuk perjalanan jarak jauh
---	---

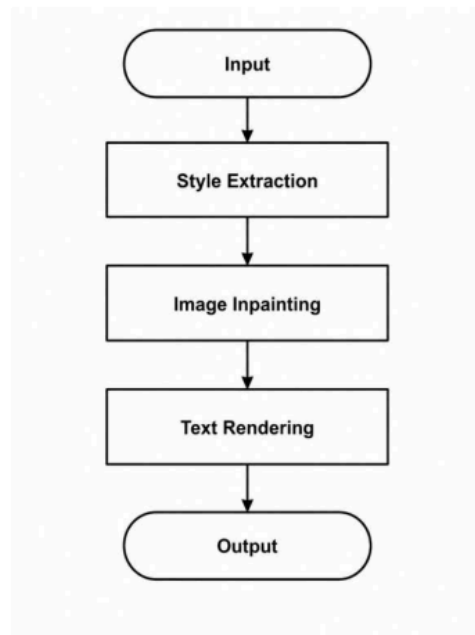
Berdasarkan hasil implementasi, modul translasi mampu menerjemahkan teks Mandarin ke dalam bahasa tujuan secara otomatis dengan tetap mempertahankan urutan hasil OCR. Penerapan mekanisme cache, batch translation, dan multithreading membantu mengurangi permintaan yang berulang ke Google Translate API sehingga proses translasi menjadi lebih efisien. Hasil terjemahan

yang diperoleh selanjutnya digunakan sebagai masukan bagi modul Image Text Swapping untuk menggantikan teks Mandarin pada gambar produk.

4.2.3.3 Implementasi Image Text Swapping

Modul *Image Text Swapping* diimplementasikan untuk menggantikan teks berbahasa Mandarin pada gambar produk dengan hasil terjemahan yang diperoleh dari modul Google Translate API. Implementasi modul ini dikembangkan menggunakan pustaka OpenCV dan Pillow (PIL) dengan menerapkan tiga tahapan utama, yaitu Style Extraction, Image Inpainting, dan Text Rendering. Pendekatan tersebut bertujuan agar hasil terjemahan dapat ditempatkan pada posisi yang sama dengan teks asli serta mempertahankan karakteristik visual gambar produk.

Selain melakukan penggantian teks, modul ini juga menyesuaikan warna tulisan, ukuran huruf, ketebalan teks, serta posisi penempatan berdasarkan karakteristik visual yang diekstraksi dari gambar asli. Dengan demikian, gambar hasil translasi tetap memiliki tata letak yang menyerupai katalog produk asli. Tahapan implementasi modul *Image Text Swapping* ditunjukkan pada Gambar 4.19.



Gambar 4.19 Alur Implementasi *Image Text Swapping*

Berdasarkan Gambar 4.19, proses *Image Text Swapping* diawali dengan menerima masukan berupa gambar asli, hasil OCR yang berisi koordinat *bounding box*, serta hasil translasi dari modul Google Translate API. Informasi tersebut digunakan sebagai dasar dalam proses penggantian teks pada gambar produk.

Tahap pertama adalah Style Extraction, yaitu proses mengekstraksi karakteristik visual dari area teks asli berdasarkan koordinat *bounding box*. Pada tahap ini sistem menganalisis warna teks, warna latar belakang, ukuran huruf, serta ketebalan tulisan sehingga karakteristik tersebut dapat digunakan kembali ketika proses penulisan teks hasil terjemahan dilakukan.

Tahap berikutnya adalah Image Inpainting, yaitu proses menghapus teks Mandarin pada gambar menggunakan metode Telea Inpainting yang tersedia pada pustaka OpenCV. Sistem membentuk *mask* berdasarkan koordinat *bounding box*, kemudian merekonstruksi area tersebut menggunakan informasi piksel di sekitarnya sehingga menghasilkan latar belakang yang siap digunakan untuk proses penempatan teks baru.

Tahap terakhir adalah Text Rendering, yaitu proses menggambar hasil terjemahan pada area yang telah dibersihkan. Implementasi tahap ini menggunakan pustaka Pillow (PIL) karena mendukung pengaturan ukuran huruf, warna teks, serta proses *font fitting* secara otomatis. Sistem menyesuaikan posisi dan ukuran teks berdasarkan dimensi *bounding box* sehingga hasil terjemahan dapat ditempatkan secara proporsional pada gambar produk. Hasil akhir dari modul ini berupa gambar katalog yang telah menggantikan teks Mandarin dengan hasil terjemahan sesuai bahasa tujuan.

Kode Program 4.17 Implementasi Style Extraction

```
class StyleExtractor:
    def extract(self, image_bgr, bbox):
        31 xs = [p[0] for p in bbox]; ys = [p[1] for p in bbox]
        x1, y1 = max(0, min(xs)), max(0, min(ys))
        x2, y2 = min(image_bgr.shape[1], max(xs)), min(image_bgr.shape[0],
max(ys))

        region = image_bgr[y1:y2, x1:x2]
        41 color = np.median(region.reshape(-1,3), axis=0)
        gray = cv2.cvtColor(region, cv2.COLOR_BGR2GRAY)
        mask = cv2.adaptiveThreshold(
            gray, 255,
            cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY_INV,
            25, 10
        )
        return TextStyle(
            font_color=(int(font[2]), int(font[1]), int(font[0])),
            bg_color=(int(bg_color[2]), int(bg_color[1]),
int(bg_color[0])),
            font_size=max(12, int((y2-y1)*0.8)),
            is_bold=is_bold
```



```
)
```

Kode Program 4.17 menunjukkan implementasi proses Style Extraction pada file `image_swapper.py`. Sistem memanfaatkan koordinat *bounding box* hasil OCR untuk mengambil area teks kemudian melakukan analisis terhadap warna latar belakang, warna huruf, ukuran teks, serta ketebalan tulisan. Informasi tersebut selanjutnya disimpan dalam objek `TextStyle` dan digunakan kembali pada proses Text Rendering agar karakteristik visual gambar tetap dipertahankan.

Kode Program 4.18 Implementasi Image Inpainting

```
def clean_text_areas(self, image, instructions):
    result = image.copy()
    for instr in instructions:
        region = result[y:y2, x:x2]
        gray = cv2.cvtColor(region, cv2.COLOR_BGR2GRAY)
        mask = cv2.threshold(
            cv2.absdiff(gray, int(np.median(gray))), 25, 255,
            cv2.THRESH_BINARY
        )[1]
        mask = cv2.dilate(mask, np.ones((3,3), np.uint8), iterations=2)
        result[y:y2, x:x2] = cv2.inpaint(
            region, mask, 3, cv2.INPAINT_TELEA
        )
    return result
```

Kode Program 4.18 menunjukkan implementasi proses Image Inpainting pada file `image_swapper.py`. Sistem membentuk *mask* berdasarkan area *bounding box* kemudian menerapkan metode Telea Inpainting yang tersedia pada pustaka OpenCV untuk merekonstruksi area yang sebelumnya berisi teks Mandarin. Hasil dari proses ini berupa latar belakang gambar yang telah dibersihkan sehingga siap digunakan pada proses penempatan teks hasil terjemahan.

Kode Program 4.19 Implementasi Text Rendering

```
def render(self, image_cv, instructions, styles):
    img_pil = Image.fromarray(
        cv2.cvtColor(image_cv, cv2.COLOR_BGR2RGB)
    ).convert("RGBA")
    overlay = Image.new("RGBA", img_pil.size, (0,0,0,0))
    draw = ImageDraw.Draw(overlay)
    for instr, style in zip(instructions, styles):
        size, font = self._fit_font(
```

```

instr.translated_text, w, h, style.is_bold
121 draw.text(
    (x+(w-tw)//2, y+(h-th)//2), instr.translated_text,
    fill=(*style.font_color,255), font=font
)

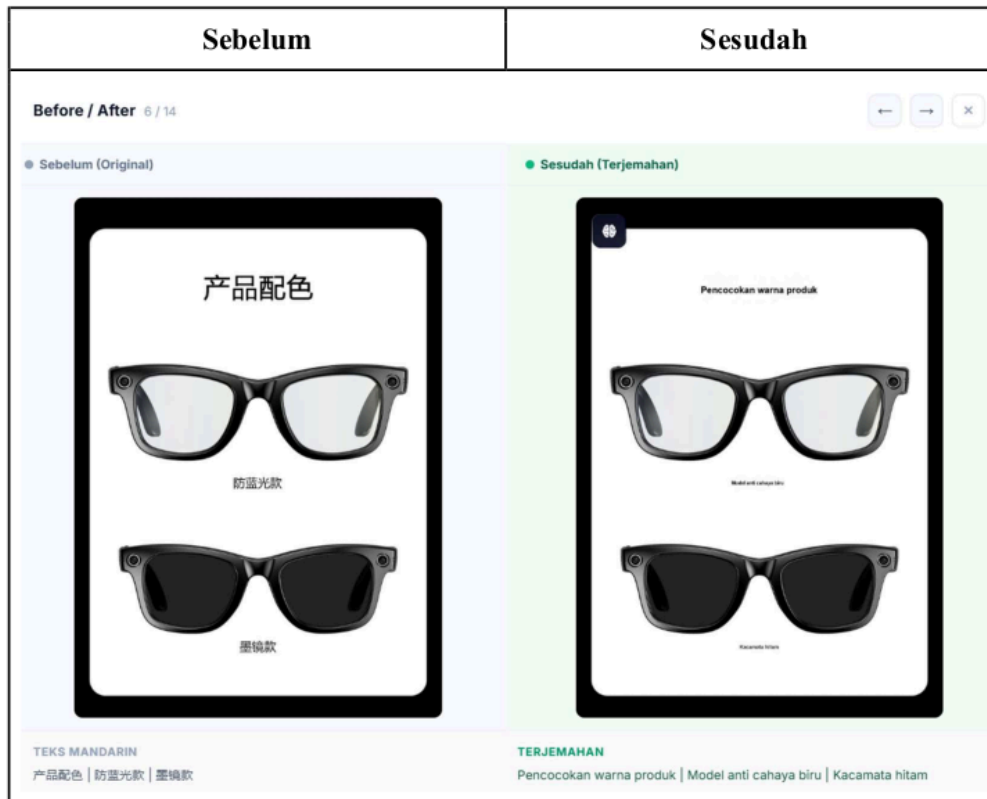
final_img = Image.alpha_composite(img_pil, overlay)
50 return cv2.cvtColor(
    np.array(final_img.convert("RGB")), cv2.COLOR_RGB2BGR
)

```

Kode Program 4.19 menunjukkan implementasi proses Text Rendering pada file `image_swapper.py` menggunakan pustaka Pillow (PIL). Sistem terlebih dahulu menyesuaikan ukuran huruf melalui mekanisme *font fitting* agar sesuai dengan dimensi *bounding box*. Selanjutnya hasil terjemahan digambar pada posisi yang sesuai menggunakan warna, ukuran, dan ketebalan huruf yang diperoleh dari proses Style Extraction. Setelah seluruh teks berhasil dirender, sistem menggabungkan lapisan teks dengan gambar hasil Image Inpainting menggunakan `Image.alpha_composite()` sehingga menghasilkan gambar katalog yang telah diterjemahkan.

Tabel 4.8 Hasil Implementasi Modul Image Text Swapping

Sebelum	Sesudah
Before / After 13 / 14 ● Sebelum (Original) TEKS MANDARIN HABEN AI Glasses 搭载 SONY 800W像素高清摄像头 图像采集 5184*3888 像素 视频采集 1080P @ 30帧/秒 ● Sesudah (Terjemahan) TERJEMAHAN Kacamata HABEN AI Dilengkapi dengan kamera SONY 8 juta piksel HD Koleksi gambar 5184*3888 piksel Pengambilan video 1080P @ 30 frame/detik	



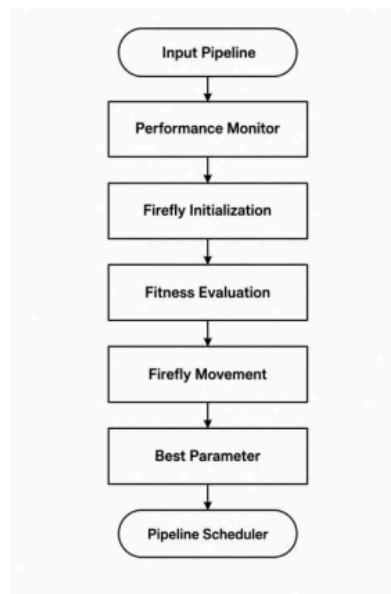
Berdasarkan hasil implementasi, modul *Image Text Swapping* mampu menghasilkan gambar katalog dengan teks berbahasa Indonesia tanpa mengubah tata letak utama pada gambar produk. Karakteristik visual seperti posisi teks, ukuran huruf, dan warna tulisan tetap dipertahankan melalui proses *Style Extraction*, *Image Inpainting*, dan *Text Rendering*, sehingga hasil translasi dapat dibaca dengan lebih baik oleh pengguna.

4.2.3.4 Implementasi Firefly Algorithm

Modul *Firefly Algorithm* diimplementasikan untuk mengoptimasi konfigurasi *pipeline scheduler* yang mengatur jumlah proses yang dapat berjalan secara bersamaan pada setiap tahapan pemrosesan gambar. Berbeda dengan implementasi *Firefly Algorithm* yang digunakan untuk optimasi pencarian solusi secara langsung, pada penelitian ini *Firefly Algorithm* digunakan untuk menentukan konfigurasi *scheduler* yang paling sesuai berdasarkan kondisi pemrosesan yang sedang berlangsung. Dengan pendekatan tersebut, proses pengunduhan gambar, OCR, translasi, dan *Image Text Swapping* dapat berjalan dengan penggunaan sumber daya yang lebih seimbang.

Implementasi *Firefly Algorithm* dijalankan secara adaptif ketika sistem mendeteksi perubahan kondisi *pipeline* melalui *Performance Monitor*. Selanjutnya algoritma mengevaluasi beberapa kandidat konfigurasi menggunakan fungsi *fitness*

untuk memperoleh konfigurasi dengan nilai *fitness* terbaik. Konfigurasi tersebut kemudian diterapkan pada *pipeline scheduler* tanpa menghentikan proses yang sedang berjalan. Tahapan implementasi Firefly Algorithm ¹³ ditunjukkan pada Gambar 4.20.



Gambar 4.20 Diagram Implementasi Firefly Algorithm

Berdasarkan Gambar 4.20, implementasi Firefly Algorithm diawali dengan menerima kondisi terkini dari pipeline scheduler. Informasi tersebut diperoleh melalui Performance Monitor yang memantau kondisi setiap tahapan pemrosesan, seperti jumlah antrean (queue), jumlah proses aktif, throughput, serta jumlah proses yang mengalami retry. Informasi tersebut digunakan sebagai dasar untuk menentukan apakah proses optimasi perlu dijalankan.

Apabila proses optimasi diperlukan, sistem melakukan Firefly Initialization dengan membangkitkan sejumlah kandidat konfigurasi *scheduler*. Setiap kandidat merepresentasikan kombinasi nilai parameter yang terdiri atas *download_limit*, *ocr_limit*, *translate_limit*, dan *swap_limit*. Parameter tersebut menentukan jumlah proses yang diizinkan berjalan secara bersamaan pada setiap tahapan pipeline.

Tahap berikutnya adalah Fitness Evaluation, yaitu proses menghitung nilai *fitness* dari setiap kandidat konfigurasi. Pada implementasi ini, fungsi *fitness* mempertimbangkan beberapa indikator kinerja pipeline, yaitu *throughput*, kesesuaian konfigurasi terhadap kondisi antrean (*parameter matching*), efisiensi penggunaan sumber daya (*resource efficiency*), serta tingkat *retry* selama proses berlangsung. Nilai *fitness* tersebut digunakan sebagai dasar dalam menentukan tingkat kecerahan (*brightness*) setiap *firefly*.

Selanjutnya dilakukan proses Firefly Movement, yaitu perpindahan posisi *firefly* menuju kandidat dengan nilai *fitness* yang lebih baik. Proses perpindahan dilakukan secara iteratif menggunakan parameter alpha, beta, dan gamma hingga diperoleh konfigurasi terbaik atau memenuhi kondisi *early stopping*. Hasil optimasi kemudian menghasilkan Best Parameter yang digunakan untuk memperbarui konfigurasi *pipeline scheduler*. Dengan pendekatan tersebut, sistem dapat menyesuaikan jumlah proses yang berjalan secara bersamaan sesuai kondisi pipeline tanpa menghentikan proses pemrosesan gambar yang sedang berlangsung.

Kode Program 4.20 Implementasi Parameter Firefly

```
@dataclass
class FAParameter:

    download_limit: int = 3
    ocr_limit: int = 2
    translate_limit: int = 3
    swap_limit: int = 2

    BOUNDS = {'download_limit': (1, 5), 'ocr_limit': (1, 2),
              'translate_limit': (1, 5), 'swap_limit': (1, 3)}

    def to_vector(self):
        return [self.download_limit, self.ocr_limit, self.translate_limit,
                self.swap_limit]

    @classmethod
    def random(cls):
        return cls(random.randint(1, 5), random.randint(1, 2),
                   random.randint(1, 5), random.randint(1, 3))
```

Kode Program 4.20 menunjukkan implementasi parameter yang dioptimasi oleh Firefly Algorithm. Setiap objek FAParameter merepresentasikan satu kandidat solusi berupa konfigurasi *Pipeline Scheduler* yang terdiri atas empat parameter. Masing-masing parameter menunjukkan jumlah maksimum *worker* yang dapat dijalankan secara bersamaan pada setiap tahapan *pipeline*. Deskripsi parameter yang digunakan dalam proses optimasi ditunjukkan pada Tabel 4.9.

Tabel 4.9 Parameter yang Dioptimasi Firefly Algorithm

Parameter	Tahapan Pipeline	Rentang Nilai
download_limit	Jumlah maksimum Download Worker yang dapat berjalan secara bersamaan.	1–5

Parameter	Tahapan Pipeline	Rentang Nilai
ocr_limit	Jumlah maksimum OCR Worker yang dapat berjalan secara bersamaan.	1–2
translate_limit	Jumlah maksimum Translation Worker yang dapat berjalan secara bersamaan.	1–5
swap_limit	Jumlah maksimum Image Text Swapping Worker yang dapat berjalan secara bersamaan.	1–3

Selain menyimpan nilai setiap parameter, kelas `FAParameter` juga mendefinisikan batas nilai (*bounds*) yang digunakan sebagai ruang pencarian (*search space*) dalam proses optimasi. Fungsi `to_vector()` digunakan untuk mengubah konfigurasi parameter ke dalam bentuk vektor sebagai masukan Firefly Algorithm, sedangkan fungsi `random()` digunakan untuk membangkitkan konfigurasi awal secara acak pada tahap inisialisasi populasi *firefly*.

Kode Program 4.21 Implementasi Fitness Function

```
def compute_fitness(snapshot, param):

    throughput = snapshot.get("throughput", 0.0)
    f_throughput = min(1.0, throughput / 2.0)
    f_param_match = ...
    f_resource = ...

    total_retry = sum(stages.get(s, {}).get("retry", 0) for s in
stage_keys)
    total_done = sum(stages.get(s, {}).get("completed", 0) for s in
stage_keys)

    retry_rate = total_retry / max(total_done, 1)
    f_retry = max(0.0, 1.0 - retry_rate)

    fitness = 0.25*f_throughput + 0.40*f_param_match + 0.20*f_resource +
0.15*f_retry

    return round(fitness, 6)
```

Kode Program 4.21 menunjukkan implementasi fungsi *fitness* pada file `firefly_optimizer.py` yang digunakan untuk mengevaluasi kualitas setiap kandidat konfigurasi. Nilai *fitness* dihitung berdasarkan empat indikator utama, yaitu *throughput*, *parameter matching*, *resource efficiency*, dan *retry rate*. Setiap

indikator memiliki bobot yang berbeda sehingga menghasilkan nilai *fitness* sebagai dasar dalam menentukan tingkat kecerahan (*brightness*) setiap firefly. Semakin tinggi nilai *fitness*, semakin baik konfigurasi tersebut dalam mengoptimalkan proses pemrosesan gambar.

Kode Program 4.22 Implementasi Adaptive Firefly Optimizer

```
def _firefly_run(self, snapshot):
    pop = [FAParameter.random() for _ in range(self.n_fireflies)]
    fits = [compute_fitness(snapshot, p) for p in pop]

    for iteration in range(self.max_iter):
        for i in range(self.n_fireflies):
            for j in range(self.n_fireflies):
                if i == j or fits[j] <= fits[i]:
                    continue

                r = self.beta_0 * math.exp(-self.gamma * r * r)
                v_i[d] = v_i[d] + beta * (v_j[d] - v_i[d]) + alpha * eps

            new_p = FAParameter.from_vector(v_i)
            new_f = compute_fitness(snapshot, new_p)

    return best_p, best_f
```

Kode Program 4.22 menunjukkan implementasi proses optimasi menggunakan Adaptive Firefly Optimizer. Secara konseptual, proses optimasi yang dilakukan terdiri atas beberapa tahapan sebagaimana ditunjukkan pada Tabel 4.10.

Tabel 4.10 Tahapan Adaptive Firefly Optimizer

Tahapan	Deskripsi
Inisialisasi Populasi	Membentuk sejumlah konfigurasi <i>worker</i> secara acak sebagai kandidat solusi awal menggunakan <code>FAParameter.random()</code> .
Evaluasi Fitness	Menghitung nilai <i>fitness</i> setiap kandidat berdasarkan kondisi <i>pipeline</i> menggunakan fungsi <code>compute_fitness()</code> .
Perbandingan Firefly	Membandingkan nilai <i>fitness</i> antar <i>firefly</i> untuk menentukan kandidat dengan kualitas solusi yang lebih baik.
Perpindahan Firefly	Memperbarui konfigurasi <i>worker</i> menuju <i>firefly</i> dengan nilai <i>fitness</i> yang lebih tinggi menggunakan persamaan Firefly Algorithm.

Tahapan	Deskripsi
Evaluasi Ulang	Menghitung kembali nilai <i>fitness</i> setelah konfigurasi diperbarui.
Seleksi Solusi Terbaik	Memilih konfigurasi dengan nilai <i>fitness</i> tertinggi sebagai hasil optimasi untuk diterapkan pada <i>Pipeline Scheduler</i> .

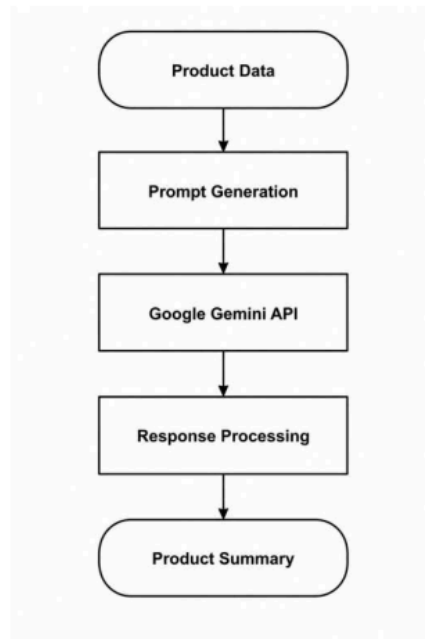
Berdasarkan tahapan tersebut, proses optimasi dilakukan secara iteratif hingga mencapai jumlah iterasi maksimum (*max_iter*). Setelah seluruh iterasi selesai, konfigurasi dengan nilai *fitness* tertinggi dipilih sebagai konfigurasi terbaik untuk memperbarui jumlah *Download Worker*, *OCR Worker*, *Translation Worker*, dan *Image Text Swapping Worker* pada *Pipeline Scheduler*.

Berdasarkan implementasi tersebut, Firefly Algorithm diintegrasikan ke dalam sistem sebagai modul yang bekerja bersama *Performance Monitor* dan *Pipeline Scheduler*. *Performance Monitor* menyediakan informasi kondisi *pipeline* sebagai masukan bagi proses optimasi, sedangkan hasil optimasi berupa konfigurasi *worker* terbaik diterapkan oleh *Pipeline Scheduler* untuk mengatur jumlah *Download Worker*, *OCR Worker*, *Translation Worker*, dan *Image Text Swapping Worker* selama proses berlangsung. Dengan mekanisme tersebut, sistem mampu menyesuaikan konfigurasi *worker* secara adaptif sesuai perubahan kondisi pemrosesan. Hasil pengujian terhadap modul Firefly Algorithm dibahas lebih lanjut pada Subbab 4.3.1.5.

4.2.3.5 Implementasi Modul Artificial Intelligence

Pada penelitian ini, Modul Artificial Intelligence diimplementasikan menggunakan Google Gemini API untuk menghasilkan ringkasan produk secara otomatis berdasarkan data hasil scraping dan OCR. Modul ini dijalankan pada sisi backend menggunakan framework Flask sehingga proses komunikasi dengan layanan Google Gemini dilakukan secara terpusat tanpa mengekspos API Key kepada Chrome Extension.

Pada implementasinya, modul menerima Product ID sebagai masukan, kemudian mengambil informasi produk dari basis data yang meliputi nama produk, harga, supplier, serta hasil OCR yang telah diterjemahkan. Informasi tersebut selanjutnya disusun menjadi sebuah prompt sebelum dikirimkan ke Google Gemini API. Hasil respons yang diterima kemudian diproses dan dikembalikan ke Chrome Extension dalam format JSON untuk ditampilkan kepada pengguna. Tahapan implementasi modul Google Gemini API ditunjukkan pada Gambar 4.21.



Gambar 4.21 Implementasi Modul Artificial Intelligence

Berdasarkan Gambar 4.21, proses diawali dengan mengambil data produk dari basis data menggunakan *Product ID* yang diterima oleh backend. Data yang diambil meliputi nama produk, harga, supplier, serta hasil OCR yang sebelumnya diperoleh dari proses ekstraksi teks pada gambar produk. Informasi tersebut digunakan sebagai sumber data untuk membangun permintaan kepada Google Gemini API.

Selanjutnya sistem melakukan Prompt Generation, yaitu menyusun instruksi yang berisi informasi produk beserta format keluaran yang diharapkan. Prompt tersebut meminta Google Gemini menghasilkan judul produk, deskripsi, spesifikasi, fitur, keunggulan, dan kegunaan produk sesuai bahasa yang dipilih pengguna. Setelah prompt selesai disusun, backend mengirimkan permintaan ke layanan Google Gemini 2.5 Flash melalui metode HTTP POST menggunakan API Key yang tersimpan pada sisi server.

Tahap berikutnya adalah Response Processing, yaitu proses membaca respons dalam format JSON yang dikembalikan oleh Google Gemini API. Backend mengekstraksi hasil deskripsi yang dihasilkan model kemudian menyusunnya ke dalam objek JSON agar dapat digunakan oleh Chrome Extension. Tahap terakhir menghasilkan Product Summary yang berisi ringkasan produk dan ditampilkan kepada pengguna sebagai informasi tambahan mengenai produk yang sedang dibuka.

Kode Program 4.22 Implementasi Google Gemini API

```

api_key = os.getenv("GEMINI_API_KEY", "")

ocr_prompt = (
    "Data produk dari 1688.com:\n"
    + "Harga: " + price_range + "\n"
    + "Teks OCR:\n" + ocr_formatted
)

gemini_body = _json.dumps({
    "contents":[{"parts":[{"text":ocr_prompt}]}],
    "generationConfig":{"temperature":0.7,"maxOutputTokens":4000}
}).encode()

gemini_req = urllib.request.Request(
    gemini_url,
    data=gemini_body,
    headers={"Content-Type":"application/json","x-goog-api-key":api_key}
)

with urllib.request.urlopen(gemini_req, timeout=45) as r:
    resp = _json.loads(r.read())

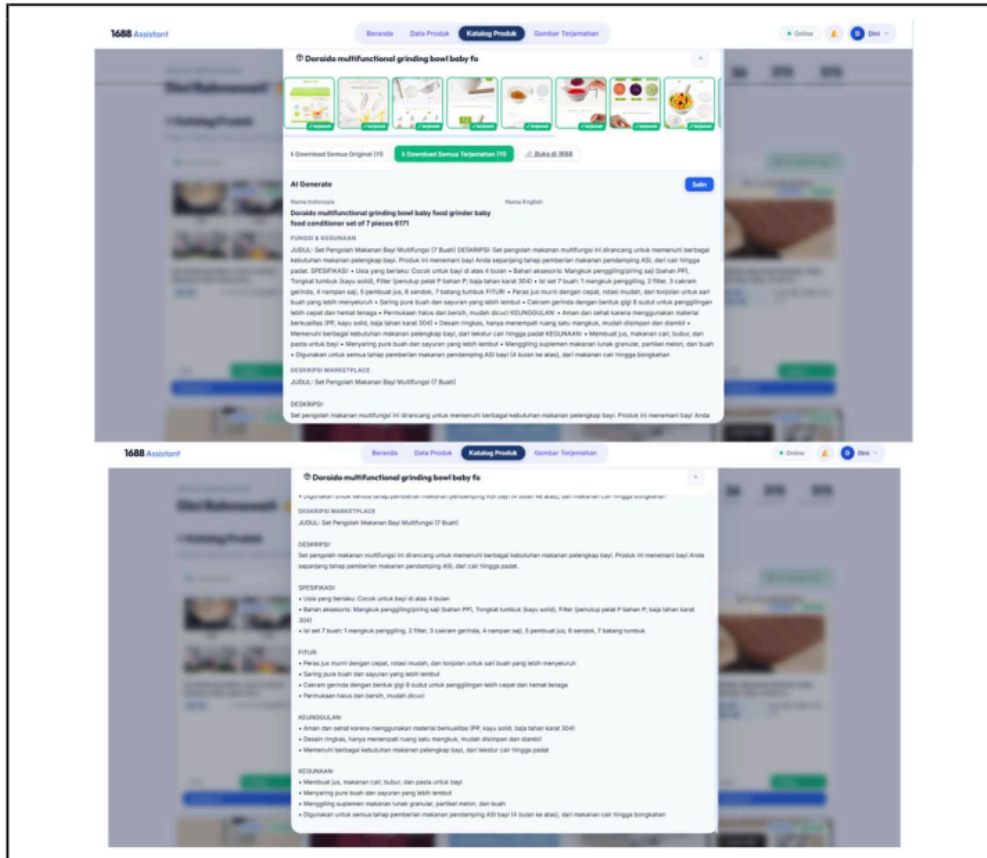
raw_text = resp["candidates"][0]["content"]["parts"][0]["text"].strip()

desc = {
    "nama_produk_id": raw_name,
    "fungsi": raw_text,
    "deskripsi_marketplace": raw_text
}

return jsonify({"success":True, "description":desc})

```

Kode Program 4.22 menunjukkan implementasi proses pembuatan ringkasan produk menggunakan Google Gemini API pada file app.py. Sistem terlebih dahulu membaca API Key yang disimpan pada konfigurasi backend untuk menjaga keamanan akses layanan. Selanjutnya backend mengambil data produk dari basis data, menggabungkan informasi hasil OCR, harga, dan informasi pendukung lainnya, kemudian menyusun data tersebut menjadi sebuah prompt yang berisi instruksi bagi model Gemini untuk menghasilkan ringkasan produk. Setelah prompt selesai disusun, backend mengirimkan permintaan HTTP POST ke layanan Google Gemini 2.5 Flash menggunakan format JSON. Permintaan tersebut memuat isi prompt serta konfigurasi model seperti temperature dan maxOutputTokens. Setelah respons diterima, sistem mengekstraksi hasil keluaran dari objek JSON kemudian membentuk objek Product Summary yang dikembalikan kepada Chrome Extension dalam format JSON. Pendekatan ini memungkinkan proses pembuatan ringkasan dilakukan secara otomatis tanpa memerlukan interaksi langsung pengguna dengan layanan Gemini API.



Gambar 4.22 Hasil *Product Summary* Pada Dashboard

Berdasarkan hasil implementasi, modul Google Gemini API berhasil menghasilkan ringkasan produk secara otomatis berdasarkan informasi hasil scraping dan OCR. Ringkasan yang dihasilkan meliputi judul produk, deskripsi, spesifikasi, fitur, keunggulan, dan kegunaan produk sesuai bahasa yang dipilih pengguna. Hasil tersebut ditampilkan pada antarmuka sistem sebagai informasi tambahan yang membantu pengguna memahami karakteristik produk tanpa harus membaca seluruh informasi asli pada halaman produk.

4.2.4 Implementasi Web Dashboard

Web Dashboard diimplementasikan sebagai media bagi pengguna untuk mengelola hasil scraping, hasil penerjemahan gambar, serta penyusunan katalog produk. Selain berfungsi sebagai penyimpanan data, dashboard juga menyediakan berbagai fitur yang mendukung proses penyusunan katalog produk multibahasa. Fitur-fitur tersebut dikelompokkan ke dalam empat menu utama pada *navigation bar*, yaitu Home, Data Produk, Katalog Produk, dan Gambar Terjemahan.

4.2.4.1 ¹³ Menu Data Produk

Menu Data Produk ⁸ digunakan untuk menampilkan seluruh data produk yang diperoleh melalui proses *scraping* pada Chrome Extension. Halaman ini berfungsi sebagai pusat pengelolaan data produk yang telah diproses oleh sistem. Informasi yang ditampilkan meliputi gambar utama produk, nama produk, status pemrosesan, tanggal pengolahan, serta informasi pendukung lainnya yang berhasil diperoleh dari platform 1688.

Selain menampilkan daftar produk, halaman ini juga menyediakan fitur pencarian untuk memudahkan pengguna menemukan produk berdasarkan nama atau kata kunci tertentu. Setiap produk ⁸ juga dilengkapi dengan tombol Detail yang dapat digunakan untuk melihat informasi produk secara lebih lengkap, termasuk hasil pengolahan gambar dan informasi produk yang telah dihasilkan oleh sistem.

Image	Product Name	Supplier	Price	Status	Date	Action
	Portable Adjustable Bedside Table, Sofa B...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail
	New Wireless Constant Temperature Milk ...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail
	High-End Led Floodlight Automatic Umbre ...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail
	Ultra-Light Fully Automatic Upfitter Black ...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail
	Outdoor Camping Portable Hanging Mochi...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail
	Manufacturer's Ready-Made Multifunction...	SHANGHAI FASHION CO., LTD.	\$5.99 - \$12.99	Pending	22 Jul 2024	Detail

Gambar 4.23 Halaman Data Produk

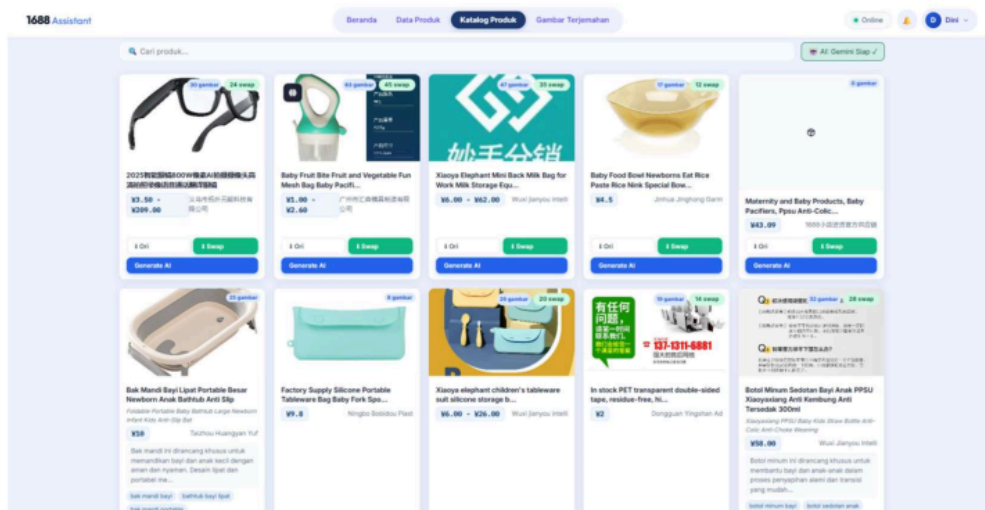
4.2.4.2 Menu Katalog Produk

Menu Katalog Produk merupakan salah satu utama pada Web Dashboard yang digunakan untuk mengelola hasil penerjemahan produk sebelum disusun menjadi katalog. Halaman ini menampilkan informasi gambar produk, hasil gambar dengan translasi, serta berbagai fasilitas pendukung penyusunan katalog.

Salah satu fitur yang tersedia pada halaman ini adalah Generate Deskripsi yang memanfaatkan Google Gemini AI. Setelah pengguna memilih produk dan menekan tombol Generate Deskripsi, sistem akan mengirimkan informasi produk yang tersimpan ke layanan Gemini AI untuk menghasilkan deskripsi produk secara otomatis. Deskripsi tersebut membantu pengguna memperoleh ringkasan informasi produk dengan cepat sehingga dapat digunakan sebagai bahan penyusunan katalog tanpa harus menulis deskripsi secara manual.

Selain itu, halaman ini juga menyediakan fitur Download Gambar Original dan Download Gambar Terjemahan sehingga pengguna dapat memperoleh gambar asli maupun gambar hasil *Image Text Swapping* secara langsung sesuai kebutuhan penyusunan katalog produk.

Tampilan halaman Katalog Produk yang memuat seluruh fitur tersebut ditunjukkan pada Gambar 4.24.

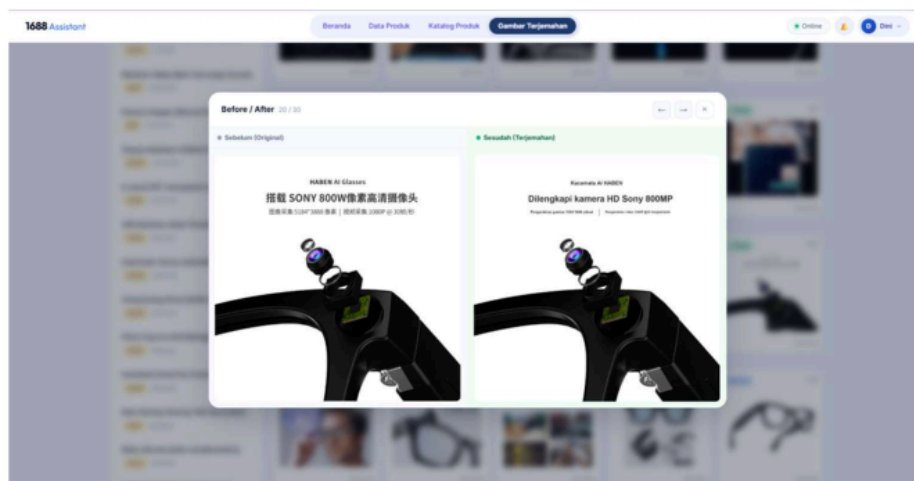


Gambar 4.24 Halaman Katalog Produk

4.2.4.3 Menu Gambar Terjemahan

Menu Gambar Terjemahan digunakan untuk menampilkan seluruh gambar yang telah berhasil diproses menggunakan modul *Image Text Swapping*. Halaman ini memudahkan pengguna untuk melihat hasil penerjemahan gambar tanpa harus membuka setiap produk secara satu per satu.

Setiap data menampilkan pasangan gambar sebelum dan sesudah proses penerjemahan sehingga pengguna dapat membandingkan hasil implementasi secara langsung. Dengan adanya halaman ini, pengguna dapat melakukan pemeriksaan visual terhadap kualitas hasil penerjemahan sekaligus memastikan bahwa tata letak dan tampilan gambar tetap terjaga setelah proses penggantian teks. Tampilan halaman Gambar Terjemahan yang menampilkan hasil sebelum dan sesudah proses penerjemahan ditunjukkan pada Gambar 4.25.



Gambar 4.25 Halaman Gambar Terjemahan

4.3 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa seluruh fungsi dan komponen yang telah diimplementasikan pada penelitian ini dapat berjalan sesuai dengan kebutuhan fungsional serta tujuan penelitian. Tahapan pengujian mengacu pada pendekatan V-Model yang telah dijelaskan pada Bab III, di mana setiap tahapan implementasi memiliki tahapan verifikasi dan validasi yang saling berpasangan. Pendekatan ini bertujuan untuk memastikan bahwa setiap modul yang dikembangkan telah memenuhi rancangan sistem sebelum diintegrasikan menjadi satu kesatuan sistem yang utuh.

Pada penelitian ini, proses pengujian dilakukan secara bertahap yang terdiri atas Unit Testing, Integration Testing, System Testing, dan User Acceptance Testing (UAT). Unit Testing digunakan untuk memverifikasi fungsi dari setiap modul sistem secara individual, meliputi modul OCR, modul translasi, modul *Image Text Swapping*, dan Firefly Algorithm. Selanjutnya, Integration Testing dilakukan untuk memastikan komunikasi dan pertukaran data antar modul dapat

berjalan dengan baik. Setelah seluruh modul berhasil diintegrasikan, System Testing dilakukan untuk mengevaluasi fungsi sistem secara keseluruhan berdasarkan kebutuhan pengguna. Tahap terakhir adalah **User Acceptance Testing (UAT)** yang bertujuan untuk mengetahui tingkat penerimaan pengguna terhadap sistem yang telah dikembangkan.

4.3.1 Unit Testing

Unit Testing dilakukan untuk memverifikasi bahwa setiap modul penyusun sistem telah berfungsi sesuai dengan kebutuhan fungsional sebelum dilakukan pengujian integrasi. Pengujian dilakukan secara terpisah terhadap setiap modul utama, yaitu modul DOM Scraping, OCR, Translation, Image Text Swapping, Artificial Intelligence, dan Firefly Algorithm. Setiap modul diuji menggunakan sejumlah data uji yang disesuaikan dengan karakteristik modul. Daftar lengkap data uji yang digunakan pada setiap pengujian disajikan pada Lampiran.

4.3.1.1 Pengujian DOM Scraping

Pengujian modul DOM Scraping dilakukan untuk memverifikasi kemampuan modul dalam mengekstraksi informasi produk dari halaman website 1688 berdasarkan struktur *Document Object Model* (DOM). Informasi yang diuji meliputi nama produk, harga, URL gambar produk, dan informasi penjual yang selanjutnya diteruskan ke backend untuk disimpan ke dalam basis data. Pengujian dilakukan menggunakan 10 halaman produk pada platform 1688 dengan kategori produk yang beragam. Daftar lengkap URL produk yang digunakan sebagai data uji disajikan pada Lampiran 2.

Keberhasilan pengujian diukur berdasarkan kemampuan modul dalam mengekstraksi seluruh atribut produk tanpa terjadi kehilangan data (*missing value*) maupun kesalahan pengambilan atribut (*attribute mismatch*). Tingkat keberhasilan (*Success Rate*) dihitung menggunakan Persamaan (4.1).

$$Success Rate = \frac{Jumlah\ Halaman\ Berhasil\ Diekstraksi}{Jumlah\ Seluruh\ Halaman\ Uji} \times 100\% \quad (4.1)$$

Tabel 4.11 Hasil Pengujian Modul DOM Scraping

Parameter	Nilai
Jumlah halaman produk diuji	10
Halaman berhasil diekstraksi	10

Halaman gagal diekstraksi	0
Success Rate	100%

Berdasarkan hasil pengujian tersebut, 10 data produk yang berhasil diekstraksi dari halaman 1688 ditampilkan pada Web Dashboard. Informasi yang berhasil diperoleh meliputi nama produk, harga produk, URL halaman, informasi penjual serta URL gambar pada Detail. Hasil tersebut menunjukkan bahwa proses ekstraksi data menggunakan Document Object Model (DOM) berhasil dilakukan dan seluruh data dapat diteruskan ke backend serta disimpan ke dalam basis data tanpa kehilangan atribut yang diperlukan. Selain mengukur tingkat keberhasilan ekstraksi halaman, dilakukan pula verifikasi terhadap setiap atribut data yang berhasil diperoleh. Hasil verifikasi atribut ³³ditunjukkan pada Tabel 4.11.

Tabel 4.12 Verifikasi Hasil Ekstraksi Data DOM

Data yang Diekstraksi	Hasil
Nama produk	Berhasil
Harga produk	Berhasil
URL gambar	Berhasil
Informasi penjual	Berhasil

Selanjutnya dilakukan pengujian relevansi URL gambar yang berhasil diekstraksi untuk memastikan bahwa seluruh gambar yang ⁹⁸diperoleh merupakan gambar katalog produk. Hasil pengujian relevansi gambar ⁹⁸ditunjukkan pada Tabel 4.12.

Tabel 4.13 Hasil Pengujian Relevansi Gambar

Parameter	Nilai
Jumlah halaman produk diuji	10
Total URL gambar yang diekstraksi	302

URL gambar katalog	290
URL gambar non-katalog (gambar profil toko)	12

Berdasarkan hasil pengujian relevansi gambar pada Tabel 4.11, modul DOM Scraping berhasil mengekstraksi 302 URL gambar dari 10 halaman produk yang diuji. Dari seluruh URL gambar yang diperoleh, 290 URL merupakan gambar katalog produk, sedangkan 12 URL lainnya merupakan gambar non-katalog. Gambar non-katalog tersebut terdiri atas gambar profil toko yang terdapat pada halaman produk serta beberapa gambar lain yang tidak termasuk ke dalam katalog produk. Hal ini menunjukkan bahwa mekanisme DOM Scraping telah berhasil mengambil seluruh elemen gambar sesuai dengan *selector* yang digunakan, namun proses penyaringan (*filtering*) terhadap gambar yang bukan bagian dari katalog produk masih perlu disempurnakan agar hanya mengekstraksi gambar yang relevan.

Berdasarkan hasil pengujian, modul DOM Scraping berhasil mengekstraksi seluruh atribut utama produk yang dibutuhkan dengan tingkat keberhasilan sebesar 100%. Nama produk, harga, URL gambar, dan informasi penjual berhasil diperoleh pada seluruh halaman produk yang diuji. Meskipun demikian, hasil pengujian relevansi gambar menunjukkan bahwa masih terdapat beberapa URL gambar non-katalog yang ikut terekstraksi sehingga mekanisme *filtering* gambar masih memerlukan penyempurnaan. Temuan tersebut tidak memengaruhi proses ekstraksi atribut utama yang digunakan pada tahapan OCR, Translation, Image Text Swapping, Artificial Intelligence, dan penyimpanan ke basis data. Dengan demikian, modul DOM Scraping dinyatakan telah berfungsi sesuai dengan kebutuhan sistem dan layak digunakan pada tahap pengujian integrasi.

4.3.1.2 Pengujian Modul OCR

Pengujian modul Optical Character Recognition (OCR) dilakukan untuk memverifikasi kemampuan modul dalam mendeteksi dan mengenali teks pada gambar produk yang diperoleh dari hasil DOM Scraping. Pengujian menggunakan 10 gambar produk dari platform 1688 yang memiliki variasi ukuran gambar, jumlah teks, warna, dan kompleksitas latar belakang. Setiap gambar diproses menggunakan EasyOCR sehingga menghasilkan beberapa *text region*. Hasil pengenalan pada setiap *text region* kemudian dibandingkan dengan teks referensi (*ground truth*) untuk menghitung tingkat akurasi OCR. Daftar lengkap data uji beserta hasil pengujian disajikan pada Lampiran 3.

Tingkat akurasi OCR dihitung menggunakan Persamaan (4.2).

$$Accuracy = \frac{Jumlah\ Text\ Region\ yang\ Dikenali\ Benar}{Juml\ \text{Seluruh text Region}} \times 100\%$$

(4.2)

Hasil pengujian modul OCR ³⁸ ditunjukkan pada Tabel 4.12.

Tabel 4.14 Hasil Pengujian Modul OCR

Parameter	Nilai
Jumlah gambar uji	10
Jumlah <i>text region</i>	81
<i>Text region</i> dikenali dengan benar	70
<i>Text region</i> tidak dikenali/salah	11
Akurasi OCR	86,41%

Untuk memberikan gambaran hasil ⁷⁸ pengujian, beberapa contoh hasil ekstraksi OCR ditunjukkan pada Tabel 4.13, sedangkan hasil pengujian secara lengkap disajikan pada Lampiran 3.

Tabel 4.15 Hasil Pengujian OCR

No	Ground Truth	Hasil OCR	Status
¹²⁶ 1	产品包装	产品包装	Benar
2	食物调理器组7件套	食物调理器组7件套	Benar
3	哆拉哆布	#@黠布	Salah
4	— (Bukan Teks)	=	Salah
5	肉	— (tidak terdeteksi)	Salah

No	Ground Truth	Hasil OCR	Status
6	— (Teks Vertikal)	— (tidak terdeteksi)	Salah

Berdasarkan hasil pengujian, modul OCR memperoleh tingkat akurasi sebesar 86,41%. Hasil tersebut menunjukkan bahwa modul OCR mampu mengenali sebagian besar teks pada gambar produk dengan baik sehingga hasil ekstraksi dapat digunakan sebagai masukan bagi modul Translation dan Image Text Swapping. Kesalahan pengenalan masih ditemukan pada beberapa *text region* dengan kualitas gambar yang rendah, kontras yang kurang baik, atau karakter yang saling berhimpitan, namun secara keseluruhan tidak memengaruhi proses pemrosesan data pada sistem.

4.3.1.3 Pengujian Pengujian Modul Translasi

Pengujian modul Translation dilakukan untuk memverifikasi kemampuan modul dalam menerjemahkan teks hasil OCR ke bahasa target menggunakan Google Translate API. Pengujian dilakukan menggunakan hasil OCR yang diperoleh dari 10 gambar produk pada platform 1688. Setiap *text region* hasil OCR dikirimkan ke Google Translate API untuk diterjemahkan ke dalam Bahasa Indonesia. Daftar lengkap data uji beserta hasil translasi disajikan pada Lampiran 4.

Pengujian dilakukan dalam dua aspek, yaitu keberhasilan proses translasi dan akurasi hasil translasi. Keberhasilan proses translasi diukur berdasarkan kemampuan sistem dalam mengirimkan permintaan (*request*) ke Google Translate API dan menerima hasil translasi tanpa mengalami kegagalan (*request failed*). Tingkat keberhasilan proses dihitung menggunakan Persamaan (4.3).

$$Success Rate = \frac{Jumlah\ Request\ Berhasil}{Jumla\ Selur\ Request} \times 100\% \quad (4.3)$$

Tabel 4.16 Hasil Pengujian Modul Translation

Parameter	Nilai
Jumlah request translasi	76
Request berhasil	76

Request gagal	0
Success Rate	100%
Rata-rata Response Time	0.398 detik

Selain mengukur keberhasilan proses translasi, dilakukan pula evaluasi terhadap kualitas hasil translasi dengan membandingkan makna hasil terjemahan terhadap teks sumber. Evaluasi dilakukan terhadap seluruh hasil translasi untuk memastikan bahwa makna hasil terjemahan tetap sesuai dengan konteks informasi produk. Tingkat akurasi translasi dihitung menggunakan Persamaan (4.4).

$$Akurasi\ Translasi = \frac{Jumlah\ Hasil\ Translasi\ Sesuai}{Jumlah\ Seluruh\ Hasil\ Translasi} \times 100\% \quad (4.4)$$

Tabel 4.17 Hasil Evaluasi Translasi

Parameter	Nilai
Jumlah hasil translasi dievaluasi	76
Hasil translasi sesuai	74
Hasil translasi kurang sesuai	2
Akurasi Translasi	97,36%

Berdasarkan hasil pengujian pada Tabel 4.17, seluruh 76 permintaan translasi berhasil diproses oleh Google Translate API sehingga diperoleh Success Rate sebesar 100% dengan rata-rata *response time* sebesar 0,398 detik. Hasil tersebut menunjukkan bahwa modul Translation mampu melakukan komunikasi dengan layanan penerjemahan secara konsisten tanpa mengalami kegagalan proses.

Tabel 4.18 Contoh Hasil Translasi

No	Teks OCR	Hasil Translasi	Evaluasi
1	产品包装	Kemasan produk	Sesuai

2	食物调理器组7件套	Set pengolah makanan 7 buah	Sesuai
3	滤出更细腻的果蔬泥	Saring puree buah dan sayuran yang lebih lembut	Sesuai
4	品名: 食物调理器组	Nama Produk: Set Kondisioner Makanan	Kurang sesuai
5	只占一个碗的位置	Itu hanya menempati ruang mangkuk	Kurang sesuai

Selanjutnya, berdasarkan hasil evaluasi kualitas translasi pada Tabel 4.18, modul Translation memperoleh akurasi translasi sebesar 97,6%. Sebagian besar hasil terjemahan telah sesuai dengan makna teks sumber sehingga dapat digunakan sebagai masukan pada proses Image Text Swapping. Namun, masih terdapat beberapa hasil translasi yang kurang sesuai dalam pemilihan diksi atau konteks kalimat. Perbedaan tersebut menunjukkan bahwa kualitas hasil translasi dipengaruhi oleh kemampuan layanan penerjemahan dalam memahami konteks kalimat serta ketepatan hasil OCR pada tahap sebelumnya.

4.3.1.4 Pengujian Modul *Image Text Swapping*

Pengujian modul *Image Text Swapping* dilakukan untuk memverifikasi kemampuan modul dalam menghapus teks asli pada gambar produk menggunakan metode *image inpainting* dan menggantinya dengan teks hasil translasi. Pengujian dilakukan menggunakan 10 gambar produk yang telah melalui proses OCR dan *Translation*. Setiap gambar diproses hingga menghasilkan gambar baru dengan teks berbahasa Indonesia. Daftar lengkap data uji beserta hasil pengujian disajikan pada Lampiran 5.

Pengujian dilakukan berdasarkan empat aspek, yaitu keberhasilan penghapusan area teks asli (*image inpainting*), penempatan teks hasil translasi, kesesuaian posisi teks terhadap area semula, dan penyimpanan gambar hasil. Tingkat keberhasilan setiap aspek dihitung menggunakan Persamaan (4.5).

Tingkat keberhasilan (*Success Rate*) dihitung menggunakan Persamaan (4.5).

$$\text{Persentase Keberhasilan Aspek} = \frac{\text{Jumlah Gambar Berhasil Pada Aspek}}{\text{Jumlah Seluruh Gambar Uji}} \times 100\% \quad (4.5)$$

Tabel 4.19 Hasil Pengujian Modul Image Text Swapping

Aspek Pengujian	Jumlah Berhasil	Persentase
Penghapusan area teks asli (<i>Image Inpainting</i>)	8 dari 10 gambar berhasil	80%
Penempatan teks hasil translasi	9 dari 10 gambar berhasil	90%
Posisi teks sesuai area semula	10 dari 10 gambar berhasil	100%
Penyimpanan gambar hasil	10 dari 10 gambar berhasil	100%

Berdasarkan hasil pengujian pada Tabel 4.19, aspek penghapusan area teks asli (*Image Inpainting*) memperoleh tingkat keberhasilan sebesar 80%, Pada 2 gambar masih ditemukan sisa teks asli yang belum terhapus secara sempurna serta warna background yang belum sesuai.

Aspek penempatan teks hasil translasi memperoleh tingkat keberhasilan sebesar 90%. Pada 1 gambar, penempatan teks kurang optimal sehingga ukuran atau tata letak teks belum sepenuhnya menyesuaikan dengan area teks asli.

Sementara itu, aspek kesesuaian posisi teks terhadap area semula dan penyimpanan gambar hasil masing-masing memperoleh tingkat keberhasilan sebesar 100%. Hal ini menunjukkan bahwa sistem mampu mempertahankan posisi teks pada area yang sesuai dengan gambar asli serta berhasil menghasilkan dan menyimpan gambar akhir tanpa mengalami kegagalan proses.

Secara keseluruhan, hasil pengujian menunjukkan bahwa modul *Image Text Swapping* memperoleh rata-rata tingkat keberhasilan sebesar 92,5% berdasarkan empat aspek pengujian, yaitu penghapusan area teks asli (*image inpainting*), penempatan teks hasil translasi, kesesuaian posisi teks terhadap area semula, dan penyimpanan gambar hasil. Hasil tersebut menunjukkan bahwa modul telah mampu menghasilkan gambar dengan teks berbahasa Indonesia yang dapat digunakan sebagai katalog produk multibahasa. Meskipun masih ditemukan beberapa keterbatasan pada proses *image inpainting* dan penempatan teks hasil translasi, hasil akhir tetap dapat dipahami dan memenuhi kebutuhan sistem.

4.3.1.5 Pengujian Modul Firefly Algorithm

Pengujian modul Firefly Algorithm bertujuan untuk mengevaluasi kemampuan algoritma dalam mengoptimalkan konfigurasi Pipeline Scheduler

sehingga waktu pemrosesan satu halaman produk dapat diminimalkan. Pengujian dilakukan menggunakan satu halaman produk pada platform 1688 yang terdiri dari delapan gambar produk. Waktu pemrosesan dihitung sejak proses pengunduhan gambar dimulai hingga seluruh gambar selesai melalui tahapan Download Image, OCR, Translation, dan Image Text Swapping.

Pada pengujian ini, Firefly Algorithm melakukan optimasi berdasarkan fungsi *fitness* yang telah dirancang pada Bab III. Fungsi *fitness* tersebut mengevaluasi kondisi *pipeline* menggunakan empat komponen, yaitu Throughput, Parameter Match, Resource Efficiency, dan Retry Rate, sehingga konfigurasi *worker* yang dipilih merupakan konfigurasi dengan nilai *fitness* terbaik.

Pengujian dilakukan dalam dua skenario. Skenario pertama menggunakan Sequential Processing sebagai kondisi awal (*baseline*) dengan konfigurasi *worker* statis. Skenario kedua menggunakan Firefly Algorithm dengan konfigurasi awal yang sama, kemudian Firefly Algorithm menyesuaikan jumlah *worker* secara dinamis berdasarkan hasil evaluasi fungsi *fitness* terhadap kondisi *pipeline* selama proses berlangsung. Perbandingan kedua skenario digunakan untuk mengetahui pengaruh optimasi Firefly Algorithm terhadap waktu penyelesaian satu halaman produk.

4.3.1.5.1 Pengujian Sequential Processing (Baseline)

Pengujian *Sequential Processing* dilakukan sebagai kondisi awal (*baseline*) sebelum diterapkannya optimasi menggunakan Firefly Algorithm. Pada pengujian ini, jumlah *worker* pada setiap tahapan *pipeline* ditetapkan masing-masing sebanyak satu *worker* sehingga konfigurasi yang digunakan adalah (1,1,1,1).

Konfigurasi tersebut menyebabkan proses berjalan secara *sequential* karena setiap tahapan hanya memiliki kapasitas untuk menangani satu *task* pada satu waktu. Dengan kapasitas *worker* yang terbatas, gambar berikutnya tidak dapat memasuki tahapan pemrosesan sebelum gambar sebelumnya menyelesaikan seluruh rangkaian proses, yaitu *Download Image*, *OCR*, *Translation*, dan *Image Text Swapping*. Akibatnya, setiap gambar diproses secara berurutan tanpa adanya pemrosesan paralel (*concurrent processing*) antar gambar.

Pendekatan tersebut digunakan sebagai kondisi dasar untuk menggambarkan proses pengolahan gambar tanpa optimasi penjadwalan. Meskipun alur pemrosesan menjadi lebih sederhana dan mudah dikendalikan, pendekatan ini belum mampu memanfaatkan sumber daya komputasi secara optimal karena setiap tahapan harus menunggu tahapan sebelumnya selesai. Akibatnya, ketika salah satu tahapan membutuhkan waktu lebih lama, tahapan lainnya tidak dapat berjalan secara bersamaan sehingga meningkatkan waktu penyelesaian keseluruhan.

Pengujian dilakukan menggunakan satu halaman produk pada web 1688.com. Hasil rata-rata waktu pemrosesan *Sequential Processing* ditunjukkan pada Tabel 4.20.

Tabel 4.20 Rata-rata Waktu Pemrosesan Sequential Processing

Tahapan	Total (detik)	Rata-rata Waktu (detik)
Download Image	6,950	0,188
OCR	188,283	5,089
Translation	4,246	0,212
Image Text Swapping	2,657	0,133
Total Keseluruhan	202,140	5,463

Berdasarkan hasil pengujian tersebut, sistem membutuhkan waktu 202,140 detik untuk menyelesaikan pemrosesan satu halaman produk. Tahapan OCR memiliki waktu pemrosesan paling tinggi sehingga menjadi proses yang paling mendominasi waktu penyelesaian keseluruhan.

4.3.1.5.2 Pengujian *Firefly Algorithm*

Setelah diperoleh hasil pengujian menggunakan *Sequential Processing* sebagai kondisi *baseline*, selanjutnya dilakukan pengujian menggunakan *Firefly Algorithm*. Pengujian ini bertujuan untuk mengetahui kemampuan *Firefly Algorithm* dalam mengoptimalkan konfigurasi *worker* pada *Pipeline Scheduler* berdasarkan kondisi pipeline yang diamati selama proses berlangsung.

Pada pengujian ini, konfigurasi awal worker dibuat sama dengan kondisi Sequential Processing, yaitu (1,1,1,1) yang masing-masing merepresentasikan jumlah Download Worker, OCR Worker, Translation Worker, dan Image Text Swapping Worker. Penggunaan konfigurasi awal yang sama bertujuan agar proses optimasi dimulai dari kondisi yang identik dengan baseline sehingga perubahan konfigurasi yang dihasilkan sepenuhnya merupakan hasil evaluasi *Firefly Algorithm*.

Selama proses pemrosesan gambar berlangsung, sistem melakukan *monitoring* terhadap kondisi pipeline secara berkala melalui Performance

Monitoring. Informasi yang diamati meliputi panjang antrean (*queue*), jumlah worker aktif, throughput, serta penggunaan sumber daya. Berdasarkan informasi tersebut, Firefly Algorithm melakukan evaluasi terhadap kondisi pipeline dan menyesuaikan konfigurasi worker apabila terdeteksi adanya bottleneck pada salah satu tahapan proses.

Hasil perubahan konfigurasi worker yang dilakukan Firefly Algorithm selama proses optimasi ditunjukkan pada Tabel 4.21.

Tabel 4.21 Hasil Optimasi Firefly Algorithm

Tahap Optimasi	Kondisi yang Terdeteksi	Konfigurasi Sebelum	Konfigurasi Sesudah	Penjelasan
Kondisi awal	Inisialisasi Pipeline	(1,1,1,1)	(1,1,1,1)	Seluruh worker menggunakan satu proses sebagai kondisi awal pengujian.
Trigger 1	Bottleneck pada Download Queue (8 antrean)	(1,1,1,1)	(5,1,1,1)	Firefly meningkatkan Download Worker untuk mempercepat proses pengambilan gambar.
Trigger 2	OCR mulai menjadi bottleneck	(5,1,1,1)	(5,2,1,1)	OCR Worker ditambah karena antrean OCR mulai meningkat.
Trigger 3	Antrean OCR tetap tinggi, Translation mulai menunggu	(5,2,1,1)	(2,2,2,1)	Download dikurangi, Translation ditambah agar pipeline lebih seimbang.
Trigger 4	Download tidak lagi menjadi bottleneck	(2,2,2,1)	(1,2,2,1)	Download Worker dikurangi sehingga sumber daya dialihkan ke proses berikutnya.
Trigger 7	Translation relatif ringan	(1,2,2,1)	(1,2,1,1)	Translation Worker dikurangi karena tidak lagi menjadi bottleneck.

Tahap Optimasi	Kondisi yang Terdeteksi	Konfigurasi Sebelum	Konfigurasi Sesudah	Penjelasan
Konfigurasi akhir	Pipeline selesai	(1,2,1,1)	(1,2,1,1)	Konfigurasi akhir yang dipilih Firefly hingga seluruh gambar selesai diproses.

Berdasarkan Tabel 4.19 dapat diketahui bahwa Firefly Algorithm memulai proses optimasi dari konfigurasi awal (1,1,1,1) yang sama dengan konfigurasi pada Sequential Processing. Pada tahap awal pemrosesan, seluruh gambar masih berada pada antrean Download sehingga Performance Monitoring mendeteksi bottleneck pada tahap tersebut. Berdasarkan hasil evaluasi Firefly Algorithm, jumlah Download Worker kemudian ditingkatkan dari 1 menjadi 5 agar proses pengunduhan dapat dilakukan secara paralel dan antrean Download dapat dikurangi.

Setelah proses pengunduhan menjadi lebih cepat, bottleneck berpindah ke tahap OCR karena jumlah gambar yang masuk ke proses OCR meningkat lebih cepat dibandingkan kemampuan OCR dalam memproses gambar. Berdasarkan kondisi tersebut Firefly kembali mengevaluasi konfigurasi worker hingga diperoleh konfigurasi akhir (1,2,1,1). Hasil ini menunjukkan bahwa Firefly mampu menyesuaikan jumlah worker secara dinamis sesuai perubahan beban kerja pada setiap tahapan pipeline.

14 Setelah proses optimasi selesai, diperoleh hasil pengujian sebagaimana ditunjukkan pada Tabel 4.22.

Tabel 4.22 Hasil Pengujian Firefly Algorithm

Parameter	Nilai
Konfigurasi Awal	(1,1,1,1)
Konfigurasi Akhir	(1,2,1,1)
Jumlah Trigger Firefly	11
Total Waktu 1 Halaman Produk	20,63 detik

Berdasarkan Tabel 4.22, seluruh gambar berhasil diproses dengan konfigurasi akhir (1,2,1,1). Firefly Algorithm melakukan sebelas kali proses evaluasi hingga diperoleh konfigurasi yang mampu menyelesaikan pemrosesan satu halaman produk dalam waktu 20,63 detik.

4.3.1.5.3 Perbandingan Sequential Processing dan Firefly Algorithm

Setelah dilakukan pengujian menggunakan Sequential Processing dan Firefly Algorithm, selanjutnya dilakukan perbandingan terhadap waktu pemrosesan satu halaman produk untuk mengetahui efektivitas optimasi yang diberikan oleh Firefly Algorithm. Tingkat pengurangan waktu pemrosesan dihitung menggunakan Persamaan (4.7).

$$\text{Pengurangan Waktu (\%)} = \frac{T_{\text{Sequential}} - T_{\text{Firefly}}}{T_{\text{Sequential}}} \times 100\% \quad (4.6)$$

dengan:

$T_{\text{Sequential}}$ = total waktu pemrosesan menggunakan Sequential Processing.

T_{Firefly} = total waktu pemrosesan menggunakan Firefly Algorithm.

Hasil perbandingan kedua metode ⁴³ ditunjukkan pada Tabel 4.21.

Tabel 4.23 Perbandingan Sequential Processing dan Firefly Algorithm

Metode	Total Waktu 1 Halaman Produk (detik)
Sequential Processing	202,140
Firefly Algorithm	20,63

Berdasarkan hasil pengujian, diperoleh:

$$\begin{aligned}
 \text{Pengurangan Waktu (\%)} &= \frac{202,140 - 20,63}{202,140} \times 100\% \\
 &= \frac{181,510}{202,140} \times 100\% \\
 &= 89,75\%
 \end{aligned}$$

Berdasarkan hasil perhitungan menggunakan Persamaan (4.6), penerapan Firefly Algorithm mampu mengurangi waktu pemrosesan sebesar 89,79% dibandingkan dengan Sequential Processing. Penurunan waktu tersebut menunjukkan bahwa mekanisme penyesuaian jumlah worker secara dinamis mampu mengurangi bottleneck pada setiap tahapan pipeline sehingga proses pemrosesan menjadi lebih efisien. Sebaliknya, pada Sequential Processing konfigurasi worker bersifat tetap sehingga sistem tidak dapat menyesuaikan diri terhadap perubahan beban kerja yang terjadi selama proses berlangsung. Oleh karena itu, Firefly Algorithm terbukti lebih efektif dalam meningkatkan efisiensi Pipeline Scheduler pada sistem yang dikembangkan.

4.3.2 Integration Testing

Integration Testing dilakukan untuk memastikan bahwa setiap modul yang telah lolos Unit Testing dapat saling berinteraksi dan bertukar data dengan benar setelah diintegrasikan menjadi satu kesatuan sistem. Pengujian dilakukan pada alur proses pengolahan informasi produk mulai dari proses pengambilan data hingga penyajian hasil pada Web Dashboard. Fokus pengujian berada pada kesesuaian aliran data (*data flow*) dan komunikasi antar modul.

Hasil pengujian integrasi ditunjukkan pada Tabel 4.24.

Tabel 4.24 Hasil Integration Testing

No	Modul yang Diintegrasikan	Skenario Pengujian	Hasil yang Diharapkan	Hasil
1	DOM Scraping → OCR	Melakukan scraping produk yang memiliki gambar berisi teks Mandarin	Gambar hasil scraping berhasil diteruskan ke modul OCR	Valid
2	OCR → Google Translate	Hasil OCR diterjemahkan	Teks hasil OCR berhasil diteruskan ke Google Translate API	Valid
3	Google Translate → <i>Image Text Swapping</i>	Hasil terjemahan digunakan untuk mengganti teks pada gambar	Gambar hasil terjemahan berhasil dihasilkan	Valid

123 No	Modul yang Diintegrasikan	Skenario Pengujian	Hasil yang Diharapkan	Hasil
4	Image Text Swapping → Gemini	Informasi produk diproses setelah translasi selesai	Gemini berhasil menghasilkan nama produk, deskripsi, dan kata kunci	Valid
5	Pipeline Scheduler → Firefly Algorithm	Pipeline dijalankan pada beberapa produk	Firefly berhasil mengoptimalkan konfigurasi scheduler	Valid
6	Seluruh Pipeline → Database	Seluruh proses pengolahan selesai	Data produk dan gambar berhasil disimpan ke basis data	Valid
7	Database → Web Dashboard	Dashboard dibuka	Informasi produk berhasil ditampilkan pada Web Dashboard	Valid

Berdasarkan hasil Integration Testing, seluruh modul berhasil saling berinteraksi sesuai dengan alur proses yang telah dirancang. Data dapat mengalir dari proses *DOM scraping*, OCR, penerjemahan, *image text swapping*, optimasi *Pipeline Scheduler* menggunakan Firefly Algorithm, hingga penyimpanan ke basis data dan penyajian melalui Web Dashboard tanpa ditemukan kesalahan integrasi.

4.3.3 System Testing

System Testing dilakukan untuk memverifikasi bahwa sistem yang telah diimplementasikan mampu memenuhi seluruh kebutuhan sistem yang telah dirumuskan pada tahap analisis kebutuhan. Pengujian dilakukan terhadap sistem secara menyeluruh sebagai satu kesatuan setelah seluruh modul berhasil diintegrasikan. Pada penelitian ini, *System Testing* mencakup pengujian fungsional menggunakan metode *Black Box Testing* serta pengujian non-fungsional untuk memastikan sistem memenuhi karakteristik operasional yang telah ditetapkan.

Pengujian fungsional dilakukan untuk memastikan bahwa setiap fungsi sistem telah berjalan sesuai dengan kebutuhan fungsional yang telah dirumuskan pada Bab III. Hasil pengujian fungsional ditunjukkan pada Tabel 4.25.

Tabel 4.25 Hasil Functional Testing

Kode	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Hasil
F-01	Pengambilan gambar produk	Pengguna melakukan import data produk dari halaman 1688	Sistem berhasil mengambil gambar produk secara otomatis	Valid
F-02	DOM Scraping	Pengguna melakukan proses scraping halaman produk	Informasi produk berhasil diperoleh dari DOM halaman	Valid
F-03	OCR	Pengguna menjalankan fitur Translate Image	Sistem berhasil mendeteksi teks Mandarin pada gambar	Valid
F-04	Google Translate	Proses translasi dijalankan	Teks hasil OCR berhasil diterjemahkan ke Bahasa Indonesia	Valid
F-05	Image Text Swapping	Proses translasi selesai	Sistem berhasil menampilkan gambar dengan teks hasil terjemahan	Valid
F-06	Gemini AI	Pengguna memilih Generate Summary	Sistem menghasilkan nama produk, deskripsi, dan kata kunci	Valid
F-07	Firefly Algorithm	Pipeline dijalankan	Sistem berhasil menjalankan optimasi konfigurasi Pipeline Scheduler	Valid
F-08	Penyimpanan Data	Proses pengolahan selesai	Data berhasil disimpan ke basis data	Valid
F-09	Web Dashboard	Pengguna membuka dashboard	Informasi produk berhasil ditampilkan	Valid
F-10	Pencarian & Filter	Pengguna mencari produk	Data berhasil ditampilkan sesuai kata kunci atau filter	Valid
F-11	Download Gambar	Pengguna memilih Download	Gambar hasil terjemahan berhasil diunduh	Valid

Berdasarkan hasil pengujian fungsional, ³⁷ seluruh fungsi utama sistem telah berjalan sesuai dengan kebutuhan yang telah ditentukan. Sistem mampu melakukan proses *scraping*, OCR, penerjemahan, *image text swapping*, optimasi *Pipeline Scheduler* menggunakan Firefly Algorithm, penyimpanan data, serta penyajian informasi melalui *Web Dashboard*.

Selanjutnya, pengujian non-fungsional dilakukan untuk memastikan bahwa sistem memenuhi kebutuhan non-fungsional yang berkaitan dengan kompatibilitas, kemudahan penggunaan, integritas data, konektivitas layanan ¹⁴⁷ eksternal, dan kebutuhan operasional sistem. Hasil pengujian non-fungsional ditunjukkan pada Tabel 4.24.

Tabel 4.26 Hasil Non-Functional Testing

Kode	Aspek	Skenario Pengujian	Hasil yang Diharapkan	Hasil
NF-01	Compatibility	Sistem dijalankan menggunakan Google Chrome	Seluruh fitur dapat berjalan dengan baik pada Google Chrome	Valid
NF-02	Usability	Pengguna mencoba seluruh fitur sistem	Antarmuka mudah dipahami dan digunakan	Valid
NF-03	Data Integrity	Pengguna melakukan <i>scraping</i> dan penyimpanan data	Seluruh data tersimpan secara konsisten tanpa kehilangan informasi	Valid
NF-04	API Integration	Sistem mengakses Google Translate API dan Gemini API	Sistem berhasil memperoleh respons dari kedua layanan	Valid
NF-05	Network	Sistem dijalankan dengan koneksi internet aktif	Seluruh proses <i>scraping</i> , translasi, dan pengolahan data berjalan dengan baik	Valid

Berdasarkan hasil pengujian non-fungsional, sistem telah memenuhi karakteristik operasional yang telah ditetapkan. ¹²² Sistem dapat berjalan dengan baik pada browser Google Chrome, mampu menjaga konsistensi data selama proses pengolahan berlangsung, berhasil terintegrasi dengan Google Translate API dan Gemini API, serta dapat menjalankan seluruh proses pengolahan informasi produk dengan baik selama koneksi internet tersedia.

⁵ 4.3.4 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) dilakukan untuk mengevaluasi tingkat penerimaan pengguna terhadap sistem yang telah dikembangkan berdasarkan

kebutuhan proses bisnis yang sebenarnya. Berbeda dengan pengujian fungsional yang berfokus pada kinerja sistem, UAT bertujuan untuk mengetahui apakah sistem mampu membantu pengguna dalam melakukan pengambilan data produk, penerjemahan gambar, serta penyusunan katalog produk multibahasa.

Pada penelitian ini, responden dipilih ⁸⁹ menggunakan teknik purposive sampling, yaitu teknik pemilihan responden berdasarkan kesesuaian peran dan pengalaman terhadap sistem yang dikembangkan. Pengujian melibatkan empat responden yang terdiri atas satu *Owner* bisnis impor, satu Staf Purchasing, satu Admin *Marketplace*, dan satu Pengguna Umum (*General User*). Tiga responden pertama dipilih karena mewakili pengguna utama (*key users*) yang secara langsung terlibat dalam proses pencarian produk, pengelolaan informasi produk, serta penyusunan katalog dari platform 1688. Sementara itu, responden Pengguna Umum ⁵ dilibatkan untuk mengevaluasi kemudahan penggunaan (*usability*) sistem dari sudut pandang pengguna yang tidak memiliki pengalaman maupun latar belakang khusus pada proses bisnis impor.

³⁴ Setelah mencoba seluruh fitur sistem, setiap responden diminta mengisi kuesioner menggunakan skala Likert lima tingkat, yaitu Sangat Tidak Setuju (1), Tidak Setuju (2), ¹⁰ Netral (3), Setuju (4), dan Sangat Setuju (5). Hasil penilaian kemudian dihitung untuk mengetahui tingkat penerimaan pengguna terhadap sistem berdasarkan pengalaman penggunaan secara langsung.

Tabel 4.27 Karakteristik Responden

No	Responden	Peran	Tanggung Jawab
1	Responden 1	Owner Bisnis Impor	Mengelola proses bisnis impor hingga distribusi barang.
2	Responden 2	Staf Purchasing	Melakukan pencarian dan pengadaan produk dari platform 1688.
3	Responden 3	Admin Marketplace	Mengelola katalog produk dan informasi produk pada marketplace.
4	Responden 4	Pengguna Umum (<i>General User</i>)	Mengevaluasi ⁵ kemudahan penggunaan sistem dari sudut pandang pengguna non-spesialis.

Pengujian User Acceptance Testing (UAT) dilakukan menggunakan delapan pernyataan yang disusun untuk mengevaluasi kemudahan penggunaan, fungsionalitas, dan manfaat sistem terhadap kebutuhan pengguna.

Tabel 4.28 Pernyataan User Acceptance Testing

No	Pernyataan
1	Sistem mempermudah proses pengambilan data produk dari platform 1688.
2	Fitur penerjemahan gambar membantu memahami informasi produk yang berbahasa Mandarin.
3	Hasil <i>Image Text Swapping</i> membuat informasi pada gambar lebih mudah dipahami.
4	Fitur <i>Generate Deskripsi</i> membantu penyusunan informasi produk untuk katalog.
5	Web <i>Dashboard</i> memudahkan pengelolaan data produk dan hasil pengolahan gambar.
6	Sistem membantu mengurangi pekerjaan manual dalam proses penyusunan katalog produk.
7	Sistem sesuai dengan kebutuhan proses bisnis impor yang dijalankan.
8	Secara keseluruhan sistem mudah digunakan dan layak diterapkan.

Persentase tingkat penerimaan pengguna dihitung menggunakan rumus berikut:

$$\text{Persentase UAT} = \frac{\text{Total Skor yang Diperoleh}}{\text{Skor Maksimum}} \times 100\% \quad (4.7)$$

Keterangan:

Total Skor yang Diperoleh = jumlah seluruh skor jawaban responden.

Skor Maksimum = jumlah responden $\times$ jumlah pertanyaan $\times$ skor tertinggi (5).

Hasil perhitungan persentase User Acceptance Testing (UAT) kemudian diinterpretasikan berdasarkan kategori tingkat penerimaan pengguna. Interpretasi persentase UAT yang digunakan dalam penelitian ini disajikan pada Tabel 4.27.

Tabel 4.29 Interpretasi Persentase UAT

Persentase	Kategori
81% – 100%	Sangat Baik
61% – 80%	Baik
41% – 60%	Cukup
21% – 40%	Kurang
0% – 20%	Sangat Kurang

Berdasarkan instrumen dan metode perhitungan yang telah ditetapkan, hasil pengujian **User Acceptance Testing (UAT)** dari responden disajikan pada Tabel 4.28.

Tabel 4.30 Hasil User Acceptance Testing

No	Pernyataan	Owner	Purchasing	Admin	General User	Total	Persentase	Kategori
1	Sistem mempermudah proses pengambilan data produk dari platform 1688.	4	5	5	5	19	95,00%	Sangat Baik
2	Fitur penerjemahan gambar membantu memahami informasi produk yang berbahasa Mandarin.	3	4	3	3	13	65,00%	Baik

N o	Pernyataan	Owner	Purchasing	Admin	General User	Total	Persentase	Kategor i
3	Hasil <i>Image Text Swapping</i> membuat informasi pada gambar lebih mudah dipahami.	4	5	5	5	19	95,00%	Sangat Baik
4	Fitur Generate Deskripsi membantu penyusunan informasi produk untuk katalog.	5	3	5	5	18	90,00%	Sangat Baik
5	Web Dashboard memudahkan pengelolaan data produk dan hasil pengolahan gambar.	3	3	5	4	15	75,00%	Baik
6	Sistem membantu mengurangi pekerjaan manual dalam proses penyusunan katalog produk.	4	5	5	5	19	95,00%	Sangat Baik
7	Sistem sesuai dengan kebutuhan proses bisnis impor yang dijalankan.	3	4	4	3	14	70,00%	Baik
8	Secara keseluruhan sistem mudah digunakan dan	4	5	5	4	18	90,00%	Sangat Baik

N o	Pernyataan	Owner	Purchasing	Admin	General User	Total	Persentase	Kategor i
	layak diterapkan.							
	Total	30	34	37	34	135	84,37%	Sangat Baik

24

Berdasarkan hasil User Acceptance Testing, sistem memperoleh tingkat penerimaan pengguna sebesar 84.37% dengan kategori Sangat Baik. Hasil tersebut menunjukkan bahwa sistem mampu membantu proses pengambilan data produk, penerjemahan gambar, penyusunan katalog, serta pengelolaan informasi produk melalui Web Dashboard. Selain memperoleh penilaian kuantitatif, responden juga memberikan masukan terhadap sistem yang dikembangkan sebagai bahan evaluasi untuk pengembangan pada penelitian selanjutnya.

BAB 5. PENUTUP

Bab ini memuat kesimpulan yang merupakan jawaban atas rumusan masalah penelitian dan saran pengembangan yang bersifat konstruktif berdasarkan temuan selama proses penelitian dan pengujian sistem.

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem, diperoleh kesimpulan sebagai berikut.

1. Berhasil dirancang dan diimplementasikan ekstensi Chrome berbasis Manifest V3 yang mampu melakukan *web scraping* data produk pada platform 1688.com secara otomatis menggunakan teknik DOM Scraping. Sistem juga terintegrasi dengan Web Dashboard untuk menyimpan dan mengelola data produk hasil *scraping* sehingga proses pengumpulan informasi produk menjadi lebih efisien. Berdasarkan hasil *User Acceptance Testing* (UAT), sistem memperoleh tingkat penerimaan pengguna sebesar 84,38% dengan kategori *Sangat Baik*, yang menunjukkan bahwa sistem telah memenuhi kebutuhan pengguna dalam proses pengumpulan dan pengelolaan data produk.
2. Berhasil diimplementasikan fitur Image Text Swapping berbasis Optical Character Recognition (OCR) yang mampu mengenali, menerjemahkan, dan mengganti teks berbahasa Mandarin pada gambar produk secara langsung. Hasil pengujian menunjukkan bahwa modul OCR memperoleh tingkat akurasi sebesar 86,41%, sedangkan modul Translation memperoleh Success Rate 100%, akurasi translasi sebesar 97,36%, serta rata-rata waktu respons 0,398 detik. Pada modul Image Text Swapping, aspek penghapusan area teks (*image inpainting*) memperoleh tingkat keberhasilan 80%, penempatan teks hasil translasi sebesar 90%, sedangkan kesesuaian posisi teks dan penyimpanan gambar hasil masing-masing mencapai 100%. Hasil tersebut menunjukkan bahwa sistem mampu menghasilkan gambar produk berbahasa Indonesia yang lebih mudah dipahami meskipun masih terdapat beberapa keterbatasan pada proses *image inpainting* dan penempatan teks.
3. Berhasil diterapkan Firefly Algorithm sebagai mekanisme optimasi pada Pipeline Scheduler untuk menyesuaikan konfigurasi *worker* secara dinamis berdasarkan kondisi *bottleneck* yang terdeteksi selama proses berlangsung. Hasil pengujian menunjukkan bahwa waktu pemrosesan satu halaman produk berkurang dari 202,140 detik pada *Sequential Processing* menjadi 20,63 detik setelah menggunakan Firefly Algorithm, atau mengalami pengurangan waktu sebesar 89,79%. Hasil tersebut menunjukkan bahwa penerapan Firefly Algorithm mampu meningkatkan efisiensi proses

penyusunan katalog produk multibahasa melalui optimasi penjadwalan pipeline.

5.2 **Saran**

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan penelitian selanjutnya adalah sebagai berikut.

1. Mengembangkan **sistem** agar **dapat** mendukung lebih banyak platform *e-commerce* selain 1688.com, seperti Alibaba, Taobao, atau Made-in-China sehingga cakupan pemanfaatan sistem menjadi lebih luas.
2. Mengembangkan sistem agar mendukung lebih banyak bahasa tujuan penerjemahan selain Bahasa Indonesia sehingga dapat dimanfaatkan oleh pengguna dari berbagai negara.
3. Mengembangkan fitur *Image Text Swapping* agar mampu mempertahankan karakteristik visual teks secara lebih baik, seperti jenis *font*, efek teks, dan tata letak yang lebih kompleks, sehingga hasil terjemahan semakin menyerupai gambar asli.

Diharapkan saran-saran tersebut dapat menjadi acuan dalam pengembangan sistem pada penelitian selanjutnya sehingga mampu meningkatkan cakupan penggunaan, kualitas hasil penerjemahan gambar, serta kinerja sistem secara keseluruhan.

DAFTAR PUSTAKA

- Alzaben, L. (2024). *The Key Differences Between Scraping Static and Dynamic Websites*. *Journal of Web Automation*, 12(3), 45–57. <https://doi.org/10.1016/j.webauto.2024.03.004>
- Chen, H., Zhang, Y., & Li, J. (2021). *Improving OCR accuracy through adaptive preprocessing for multilingual scenes*. *Pattern Recognition Letters*, 145, 12–20. <https://doi.org/10.1016/j.patrec.2021.04.007>
- DeepMind Research. (2024). *Gemini 1.5 Technical Report: A Multimodal Large Language Model*. Google DeepMind. <https://deepmind.google/discover/blog/>
- DMAME Journal. (2022). *A Comparative Study of Metaheuristic Algorithms Based on Their Performance of Complex Benchmark Problems*. *Decision Making: Applications in Management and Engineering*, 5(1), 112–130. <https://doi.org/10.31181/dmame2205112>
- FASDANA. (2025). *Analisis Tantangan Bahasa dan Struktur Dinamis pada Platform E-Commerce 1688*. *Jurnal Sistem Informasi Global*, 8(2), 88–96. <https://doi.org/10.31145/jsig.v8i2.2241>
- Google Chrome for Developers. (2024). *Manifest V3 Overview*. Google Developer Documentation. <https://developer.chrome.com/docs/extensions/mv3/>
- Gabor, D., Singh, R., & Mistry, A. (2018). *Noise reduction and binarization for document image enhancement*. *International Journal of Computer Vision Applications*, 29(4), 301–312. <https://doi.org/10.1007/s11263-018-1134-9>
- Huang, X., Li, P., & Yang, T. (2023). *Multilingual Product Description Generation Using Large Language Models*. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. <https://doi.org/10.18653/v1/2023.acl-main.412>
- Lam, C. (2021). *Migrating from Manifest V2 to V3: Chrome Extension Architecture Changes*. *Google Developers Blog*. <https://developer.chrome.com/docs/extensions/mv3/mv2-sunset/>
- Liu, Z., Chen, F., & Wang, X. (2022). *Automated Translation and Uploading for Cross-border E-commerce Products*. *Journal of Applied Computing Research*, 10(2), 221–232. <https://doi.org/10.1016/j.jacr.2022.02.008>

Mahariani, N. (2022). *Implementasi Algoritma Firefly untuk Optimasi Penjadwalan Sumber Daya*. *Jurnal Teknologi Informasi dan Komputer (JTik)*, 9(1), 55–64. <https://doi.org/10.33322/jtik.v9i1.1776>

Melyawati, S., Rahman, D., & Putri, A. (2024). *Comparison of Automation Testing on Card Printer Project Using Playwright and Selenium Tools*. *International Journal of Software Testing*, 7(2), 133–142. <https://doi.org/10.1016/j.ijsoftest.2024.02.006>

Putra, A. R., Sari, D., & Handoko, P. (2022). *Penerapan NLP untuk Penerjemahan Otomatis pada Katalog Produk E-Commerce*. *Jurnal Nasional Informatika dan Komputer (JURNIKOM)*, 9(4), 78–85. <https://doi.org/10.33322/jurnikom.v9i4.211>

Shetty, R., Kumar, P., & Singh, A. (2025). *A Unified Flask-Based Framework for Image Text Recognition, Multilingual Translation, and Text Summarization*. *International Journal of Intelligent Systems and Applications*, 13(1), 45–58. <https://doi.org/10.1016/j.ijisa.2025.01.004>

Suryana, D., Lestari, R., & Prasetyo, I. (2024). *Perancangan Ekstensi Chrome untuk Otomatisasi Tugas Web*. *Jurnal Nasional Teknologi Informasi (JNTI)*, 6(3), 210–219. <https://doi.org/10.33322/jnti.v6i3.253>

Wang, L., Zhou, M., & Lin, H. (2020). *Hybrid model for Chinese character recognition based on Tesseract-OCR*. *Neural Computing and Applications*, 32(15), 11245–11256. <https://doi.org/10.1007/s00521-020-04917-6>

Yang, X., Chen, M., & Xu, H. (2020). *SwapText: Image-Based Texts Transfer in Scenes*. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 14700–14709. <https://doi.org/10.1109/CVPR42600.2020.01471>

Yang, X. S. (2010). *Firefly Algorithm, Stochastic Test Functions and Design Optimization*. *International Journal of Bio-Inspired Computation*, 2(2), 78–84. <https://doi.org/10.1504/IJBIC.2010.032124>

Alsadie, D., & Alsulami, M. (2024). *Enhancing workflow efficiency with a modified Firefly Algorithm for hybrid cloud edge environments*. *Scientific Reports*, 14, 24675.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide* (2nd ed.). Addison-Wesley.

Gemini Team, Google. (2023). *Gemini: A Family of Highly Capable Multimodal Models*. arXiv preprint arXiv:2312.11805. <https://arxiv.org/abs/2312.11805> Khder, M. A. (2021). *Web Scraping or Web Crawling: State of Art, Techniques,*

Approaches and Application. *International Journal of Advances in Soft Computing and its Applications*, 13(3), 144–168.

OpenStax. (2024). *Introduction to Computer Science*. OpenStax, Rice University. <https://openstax.org/books/introduction-computer-science>

Yang, X. S. (2008). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe &

T. Zeugmann (Eds.), *Stochastic Algorithms: Foundations and Applications (SAGA 2009)*,

Lecture Notes in Computer Science, Vol. 5792 (pp. 169–178). Springer https://doi.org/10.1007/978-3-642-04944-6_14

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.

Kroch, M. (2021). *Comparative analysis of Python web frameworks for machine learning API deployment*. *Journal of Software Engineering and Applications*, 14(6), 210–225.

Ronacher, A. (2020). *Flask Documentation Release 2.0.x*. Pallets Projects.

Deelman, E., Gorgan, J., & Malawski, M. (2019). *Workflow management and scheduling in complex multi-stage pipelines*. *Future Generation Computer Systems*, 94, 820–835.

Klosowski, P., Zabierowski, W., & Napieralski, A. (2022). *Concurrency and scheduling optimization in asynchronous microservice architectures*. *IEEE Access*, 10, 41200–41212.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). John Wiley & Sons.

Chandra, S., & Tiwari, A. (2021). *Performance evaluation of dynamic web scraping techniques on JavaScript-rendered applications*. *Journal of Web Engineering*, 20(3), 745–768.

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7th ed.). O'Reilly Media.

Varghese, R., & Vijay, M. (2019). *DOM-based data extraction for modern dynamic web applications*. *International Journal of Computer Applications*, 178(25), 12–18.

W3C. (2020). *World Wide Web Consortium: Document Object Model (DOM) Technical Specification*. W3C Recommendation.

Chandra, S., & Tiwari, A. (2021). *Performance evaluation of dynamic web scraping techniques on JavaScript-rendered applications*. *Journal of Web Engineering*, 20(3), 745–768.

Chen, X., Zhang, Y., & Liu, H. (2022). *An integrated pipeline for visual text extraction and seamless image text replacement*. *IEEE Transactions on Image Processing*, 31, 4120–4132.

Klosowski, P., Zabierowski, W., & Napieralski, A. (2022). *Concurrency and scheduling optimization in asynchronous microservice architectures*. *IEEE Access*, 10, 41200–41212.

Lam, A. (2021). *Building Chrome Extensions with Manifest V3*. Apress.

Zhao, L., & Wang, J. (2023). *Automated product catalog localization system using web extraction and optical character recognition*. *Journal of E-Commerce Research and Applications*, 58, 101245.

Turnitin Dini

ORIGINALITY REPORT

11%

SIMILARITY INDEX

10%

INTERNET SOURCES

5%

PUBLICATIONS

6%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Sangga Buana YPKP	1%
	Student Paper	
2	repository.its.ac.id	1%
	Internet Source	
3	Submitted to Universitas Putera Batam	<1%
	Student Paper	
4	repository.ub.ac.id	<1%
	Internet Source	
5	lab.tik.pnj.ac.id	<1%
	Internet Source	
6	dspace.uui.ac.id	<1%
	Internet Source	
7	Submitted to Universitas Brawijaya	<1%
	Student Paper	
8	Submitted to UIN Raden Intan Lampung	<1%
	Student Paper	
9	Submitted to UPN Veteran Yogyakarta	<1%
	Student Paper	
10	repository.nusamandiri.ac.id	<1%
	Internet Source	
11	journals.upi-yai.ac.id	<1%
	Internet Source	
12	Aulia Cahyaningrum, Agus Yulianto. "Perancangan Sistem Informasi E-commerce dan Branding Digital PT. Trimitra Surya Cemerlang Menggunakan Metode Prototype", remik, 2025	<1%
	Publication	
13	repository.dinamika.ac.id	<1%
	Internet Source	
14	Submitted to Universitas Diponegoro	<1%
	Student Paper	

15	eprints.upj.ac.id Internet Source	<1 %
16	emmanuelorozco.com Internet Source	<1 %
17	repository.ar-raniry.ac.id Internet Source	<1 %
18	repo.darmajaya.ac.id Internet Source	<1 %
19	id.123dok.com Internet Source	<1 %
20	Arsa Anggun Anggraeni, Felista Argita Putri, Salsabila Mutiara Defa, Fajar Mahardika, Nur Muniroh, Cahya Vikasari. "Pengembangan Sistem Informasi Petshop Pawarca Berbasis Web Menggunakan Metode Rapid Application Development (RAD) dan Black Box Testing", INFOKABIN (Informatika Komputasi Aplikasi dan Bisnis), 2026 Publication	<1 %
21	www.scribd.com Internet Source	<1 %
22	products.documentprocessing.com Internet Source	<1 %
23	repository.unsil.ac.id Internet Source	<1 %
24	Submitted to Politeknik Manufaktur Negeri Bangka Belitung Student Paper	<1 %
25	Submitted to University of Bedfordshire Student Paper	<1 %
26	repository.unissula.ac.id Internet Source	<1 %
27	repository.usni.ac.id Internet Source	<1 %
28	text-id.123dok.com Internet Source	<1 %
29	Anakagung Agung. "SISTEM INVENTARIS DAN PROMOSI TOKO BAJU LOVE LACE GIANYAR", Jurnal Informatika dan Teknik Elektro Terapan, 2026 Publication	<1 %

Submitted to East Union High School

30	Student Paper	<1 %
31	Submitted to University of San Diego Student Paper	<1 %
32	jurnal.bsi.ac.id Internet Source	<1 %
33	digilib.uinsby.ac.id Internet Source	<1 %
34	ojs.uajy.ac.id Internet Source	<1 %
35	Submitted to Universitas Satya Negara Indonesia Student Paper	<1 %
36	blog.huli.tw Internet Source	<1 %
37	ejurnal.poliban.ac.id Internet Source	<1 %
38	adoc.pub Internet Source	<1 %
39	etheses.uin-malang.ac.id Internet Source	<1 %
40	eprints.utar.edu.my Internet Source	<1 %
41	www.web-dev-qa-db-fra.com Internet Source	<1 %
42	Muhammad Afif Hidayat, Idham Fatirrohman Syufi. "Implementasi Sistem E-Commerce Berbasis Web Menggunakan Laravel pada PT. Saveta", YASIN, 2026 Publication	<1 %
43	irmajhe.blogspot.com Internet Source	<1 %
44	journal.uin-alauddin.ac.id Internet Source	<1 %
45	Submitted to McNeese State University Student Paper	<1 %
46	Submitted to Politeknik Penerbangan Surabaya Student Paper	<1 %

47	Submitted to Sekolah Teknik Elektro & Informatika Student Paper	<1 %
48	huggingface.co Internet Source	<1 %
49	repository.mediapenerbitindonesia.com Internet Source	<1 %
50	webs-deim.urv.cat Internet Source	<1 %
51	Submitted to Konsorsium PTS Batch 5 Student Paper	<1 %
52	Submitted to Universitas Khairun Student Paper	<1 %
53	Zalfaa Astiad Ryada, Apriade Voutama. "Perancangan Sistem Informasi Booking Lapangan Badminton Berbasis Web Menggunakan Unified Modeling Language", RIGGS: Journal of Artificial Intelligence and Digital Business, 2026 Publication	<1 %
54	ejurnal.undipa.ac.id Internet Source	<1 %
55	lib.ibs.ac.id Internet Source	<1 %
56	repository.polman-babel.ac.id Internet Source	<1 %
57	repository.umy.ac.id:8080 Internet Source	<1 %
58	Submitted to (school name not available) Student Paper	<1 %
59	Ratih Haryati, Muhammad Abdul Hamid, Nur Hasan. "Tathwir Mâddah Qawa'id Al Nahw Bi Istikhdam Al Wasâit Articulate Storyline Fi al Madrasah al Mutawassitah Al Islâmiyyah", Al Mi'yar: Jurnal Ilmiah Pembelajaran Bahasa Arab dan Kebahasaaraban, 2025 Publication	<1 %
60	Submitted to UPA Perpustakaan UNTIRTA Student Paper	<1 %
61	Submitted to Universitas Dian Nuswantoro Student Paper	<1 %

62	Internet Source	<1 %
63	jacis.pubmedia.id Internet Source	<1 %
64	Submitted to Admin Turnitin UIIS Student Paper	<1 %
65	Submitted to Boston University Student Paper	<1 %
66	Submitted to IUBH - Internationale Hochschule Bad Honnef-Bonn Student Paper	<1 %
67	Submitted to Universitas Islam Riau Student Paper	<1 %
68	arunbaby.com Internet Source	<1 %
69	repository.upnjatim.ac.id Internet Source	<1 %
70	Submitted to Hult International Business School Student Paper	<1 %
71	Submitted to Universitas Dharmawangsa Student Paper	<1 %
72	Submitted to University of Glasgow Student Paper	<1 %
73	cdn.juris.id Internet Source	<1 %
74	repository.untag-sby.ac.id Internet Source	<1 %
75	Submitted to City University of Seattle Student Paper	<1 %
76	Submitted to Telkom University Student Paper	<1 %
77	Thilagavathy R, Veeramani T, B. Sundaravadivazhagan, Deebalakshmi R. "Evolution of software engineering: From traditional to modern approaches", Elsevier BV, 2026 Publication	<1 %
78	Submitted to Universitas Negeri Jakarta Student Paper	<1 %

79	Submitted to Universitas Wijaya Kusuma Surabaya Student Paper	<1 %
80	Zainudin Husain, Saprina Mamase, Irwan Karim, Basri Umar. "Sistem Informasi Rantai Pasok dan Distribusi Kebutuhan Penjualan Berbasis Web", Jurnal Pengembangan Teknologi Informasi dan Komunikasi (JUPTIK), 2026 Publication	<1 %
81	core.ac.uk Internet Source	<1 %
82	repository2.unw.ac.id Internet Source	<1 %
83	staging.sanfordfl.gov Internet Source	<1 %
84	www.kabarsumbawa.com Internet Source	<1 %
85	1library.net Internet Source	<1 %
86	Submitted to Universitas Muhammadiyah Purwokerto Student Paper	<1 %
87	journal.stmik-bandung.ac.id Internet Source	<1 %
88	library.binus.ac.id Internet Source	<1 %
89	prosiding.aritekin.or.id Internet Source	<1 %
90	El-Zonkoly, Amany, and Leandro dos Santos Coelho. "Optimal allocation, sizing of PHEV parking lots in distribution system", International Journal of Electrical Power & Energy Systems, 2015. Publication	<1 %
91	Roys Pakaya, Abdul Rahman Tapate, Salman Suleman. "PERANCANGAN APLIKASI PENJUALAN HEWAN TERNAK UNTUK QURBAN DAN AQIQAH DENGAN METODE UNIFIED MODELING LANGUAGE (UML)", Jurnal Technopreneur (JTech), 2020 Publication	<1 %
92	Submitted to UIN Sultan Syarif Kasim Riau Student Paper	<1 %
93	Submitted to Universitas Muria Kudus Student Paper	<1 %

94	app.studyraid.com Internet Source	<1 %
95	b2bleadfinder.io Internet Source	<1 %
96	bapeten.go.id Internet Source	<1 %
97	binus.ac.id Internet Source	<1 %
98	dokumen.tips Internet Source	<1 %
99	dspace.vut.cz Internet Source	<1 %
100	enjiner.com Internet Source	<1 %
101	eprints.universitaspurabaya.ac.id Internet Source	<1 %
102	ijcs.stmikindonesia.ac.id Internet Source	<1 %
103	journal.amikveteran.ac.id Internet Source	<1 %
104	journal.lembagakita.com Internet Source	<1 %
105	jurnal.researchideas.org Internet Source	<1 %
106	kc.umn.ac.id Internet Source	<1 %
107	moam.info Internet Source	<1 %
108	python.hotexamples.com Internet Source	<1 %
109	rdr.io Internet Source	<1 %
110	repository.usu.ac.id Internet Source	<1 %

111

Internet Source

<1 %

112

Faris Ihsan Setianto, Muhammad Jagad Satriyo, Arif Rahman Hakim, Nanang Nanang. "Pengembangan Sistem Informasi Katalog Obat dan Pemesanan Online Apotek HEHA", Indonesian Journal of Multidisciplinary on Social and Technology, 2026

Publication

<1 %

113

Ivan Malashin, Igor Masich, Vadim Tynchenko, Andrei Gantimurov, Vladimir Nelyub, Aleksei Borodulin. "Image Text Extraction and Natural Language Processing of Unstructured Data from Medical Reports", Machine Learning and Knowledge Extraction, 2024

Publication

<1 %

114

Submitted to Napier University

Student Paper

<1 %

115

Submitted to Universitas Muhammadiyah Palembang

Student Paper

<1 %

116

digilib.itb.ac.id

Internet Source

<1 %

117

digilib.uinkhas.ac.id

Internet Source

<1 %

118

docslide.us

Internet Source

<1 %

119

eprints.pancabudi.ac.id

Internet Source

<1 %

120

eprints.unsri.ac.id

Internet Source

<1 %

121

ismavatar.com

Internet Source

<1 %

122

j-ptiik.ub.ac.id

Internet Source

<1 %

123

jurnal.muaraedukasi.id

Internet Source

<1 %

124

rama.uniku.ac.id

Internet Source

<1 %

125

repository.unwira.ac.id

Internet Source

<1 %

126	studentcompetitioin.meiguoxingren.com.cn Internet Source	<1 %
127	techbusiness.it Internet Source	<1 %
128	vida.id Internet Source	<1 %
129	www.blogsederhana.web.id Internet Source	<1 %
130	Khabat Khosravi, Mahdi Panahi, Dieu Tien Bui. "A comprehensive study of new hybrid models for Adaptive Neuro-Fuzzy Inference System (ANFIS) with Invasive Weed Optimization (IWO), Differential Evolution (DE), Firefly (FA), Particle Swarm Optimization (PSO) and Bees (BA) algorithms for spatial prediction of groundwater spring potential mapping", Copernicus GmbH, 2018 Publication	<1 %
131	Muhammad Sani, Muhammad Asep Subandri. "Analisis Kerentanan Website Desa Sekecamatan Bengkalis Menggunakan Vulnerability Assessment", Jurnal Teknik Industri Terintegrasi, 2025 Publication	<1 %
132	Nurrizkyta Aulia Hanifah, Muhammad Aqil Musthafa Arrachman Musthafa Arrachman, Muhammad Fillah Al Fatih, Nasywa Shafa Salsabila et al. "PEMBUATAN PLATFORM DIGITAL UNTUK EDUKASI DAN E-COMMERCE HIDROPONIK BERBASIS BUSINESS INTELLIGENCE: STUDI KASUS HYDROSPACE", Djtechno: Jurnal Teknologi Informasi, 2025 Publication	<1 %
133	Roys Pakaya. "RANCANG BANGUN APLIKASI MANAJEMEN BARBERSHOP UNTUK OPTIMASI BOOKING DAN LAPORAN KEUANGAN", Journal Of Software Engineering And Communication, 2026 Publication	<1 %
134	calmops.com Internet Source	<1 %
135	conference.upnvj.ac.id Internet Source	<1 %
136	dpa.bulungan.go.id Internet Source	<1 %
137	dspace.ups.edu.ec Internet Source	<1 %
138	ejournal.techcart-press.com Internet Source	<1 %

139	ejurnal.provisi.ac.id Internet Source	<1 %
140	jurnalmahasiswa.com Internet Source	<1 %
141	nurtyasrini.blogspot.com Internet Source	<1 %
142	ojs.unigal.ac.id Internet Source	<1 %
143	publikasi.hawari.id Internet Source	<1 %
144	repository.usu.ac.id Internet Source	<1 %
145	repository.itpln.ac.id Internet Source	<1 %
146	repository.nusaputra.ac.id Internet Source	<1 %
147	repository.ppns.ac.id Internet Source	<1 %
148	repository.teknokrat.ac.id Internet Source	<1 %
149	repository.uinsaizu.ac.id Internet Source	<1 %
150	repository.umsu.ac.id Internet Source	<1 %
151	repository.unjaya.ac.id Internet Source	<1 %
152	wirebiru.blogspot.com Internet Source	<1 %
153	www.researchgate.net Internet Source	<1 %
154	Submitted to The Robert Gordon University Student Paper	<1 %
155	hdl.handle.net Internet Source	<1 %

156	Erlin Dwi Haryanti, Apriade Voutama, Siska Siska. "Perancangan Model Implementasi Sistem ERP Berbasis Odoo untuk UMKM Apotek", RIGGS: Journal of Artificial Intelligence and Digital Business, 2026 Publication	<1 %
157	Rizki Maulana, Hari Setiaji. "PENGEMBANGAN MODUL KASIR BERBASIS WEB-ANDROID PADA SISTEM POS FULL SELF-SERVICE", Technologia : Jurnal Ilmiah, 2026 Publication	<1 %
158	dev.to Internet Source	<1 %
159	repository.usd.ac.id Internet Source	<1 %
160	Ika Nurlaili Isnainiyah, lin Ernawati. "Optimalisasi Penggunaan Website Kabupaten Pandeglang untuk Penyebaran Informasi Publik", Jurnal Pengabdian kepada Masyarakat UBJ, 2021 Publication	<1 %
161	datahacker.rs Internet Source	<1 %

Exclude quotes On
Exclude bibliography On

Exclude matches Off