Jurnal Multidisiplin Indonesia (JOUMI)
Volume 4 Nomor 3 September 2026
E-ISSN : 2986-7541; P-ISSN : 2986-7533, Hal 126-148
DOI: https://doi.org/10.62007/joumi.v4i3.904
Tersedia: https://ejournal.alhafiindonesia.co.id/index.php/JOUMI

Perancangan Ekstensi Chrome Berbasis Firefly Algorithm untuk Optimasi Scraping dan *Image Text Swapping* pada Platform 1688

Dini Rahmawati[1*], Tjahjanto[2], Noor Falih[3]
[1-3] Universitas Pembangunan Nasional Veteran Jakarta,Indonesia

[*]*Penulis Korespondensi: dinirahmaw23@gmail.com*

Abstract. *Indonesian import business owners widely rely on 1688.com as a primary sourcing platform for products. However, most product information, including specifications embedded within images (in-image text), is presented in Mandarin and cannot be translated effectively using conventional browser translation tools. This limitation makes product evaluation and decision-making slow, inefficient, and highly dependent on manual translation. This research aims to develop 1688 Assistant, a Chrome extension that automates product data extraction, translates image-based text through Optical Character Recognition (OCR) and image text replacement, and optimizes the scraping and image-processing workflow using the Firefly Algorithm within the Pipeline Scheduler. The study adopts a Research and Development (R&D) methodology based on the V-Model, encompassing requirement analysis, system design, implementation, verification, and validation. System performance was evaluated through Unit Testing, Integration Testing, System Testing, and User Acceptance Testing (UAT). The testing results indicate that all functional modules operated successfully without critical errors, confirming the reliability and stability of the developed application. Furthermore, the UAT achieved a score of 84.16%, categorized as Very Good, indicating a high level of user satisfaction in terms of usability, functionality, and efficiency. The developed extension enables import business owners to understand product information on 1688.com more quickly, accurately, and efficiently, thereby supporting better purchasing decisions and improving overall business productivity.*

Keywords*: Chrome Extension; Firefly Algorithm; Image Text Swapping; Optical Character Recognition; Web Scraping.*

Abstrak. Para pelaku bisnis impor di Indonesia banyak mengandalkan 1688.com sebagai platform utama untuk mencari produk. Namun, sebagian besar informasi produk, termasuk spesifikasi yang tertanam dalam gambar (teks dalam gambar), disajikan dalam bahasa Mandarin dan tidak dapat diterjemahkan secara efektif menggunakan alat terjemahan browser konvensional. Keterbatasan ini membuat evaluasi produk dan pengambilan keputusan menjadi lambat, tidak efisien, dan sangat bergantung pada terjemahan manual. Penelitian ini bertujuan untuk mengembangkan 1688 Assistant, sebuah ekstensi Chrome yang mengotomatiskan ekstraksi data produk, menerjemahkan teks berbasis gambar melalui Optical Character Recognition (OCR) dan penggantian teks gambar, serta mengoptimalkan alur kerja scraping dan pemrosesan gambar menggunakan Algoritma Firefly dalam Pipeline Scheduler. Studi ini mengadopsi metodologi Penelitian dan Pengembangan (R&D) berdasarkan Model V, yang mencakup analisis kebutuhan, desain sistem, implementasi, verifikasi, dan validasi. Kinerja sistem dievaluasi melalui Pengujian Unit, Pengujian Integrasi, Pengujian Sistem, dan Pengujian Penerimaan Pengguna (UAT). Hasil pengujian menunjukkan bahwa semua modul fungsional beroperasi dengan sukses tanpa kesalahan kritis, yang menegaskan keandalan dan stabilitas aplikasi yang dikembangkan. Selain itu, UAT mencapai skor 84,16%, yang dikategorikan sebagai Sangat Baik, menunjukkan tingkat kepuasan pengguna yang tinggi dalam hal kegunaan, fungsionalitas, dan efisiensi. Ekstensi yang dikembangkan memungkinkan pemilik bisnis impor untuk memahami informasi produk di 1688.com dengan lebih cepat, akurat, dan efisien, sehingga mendukung pengambilan keputusan pembelian yang lebih baik dan meningkatkan produktivitas bisnis secara keseluruhan.

Kata kunci: Algoritma *Firefly*; Ekstensi Chrome; Pengambilan Data Web; Pengenalan Karakter Optik; Pertukaran Teks Gambar.

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

Naskah Masuk: 29 Mei 2026; Revisi: 30 Juni 2026; Diterima: 31 Juli 2026; Terbit 01 September 2026

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 berdampak langsung pada proses analisis produk (product analysis), tahap krusial sebelum pelaku usaha impor mengambil keputusan pembelian, karena spesifikasi dan keunggulan produk yang disajikan dalam bentuk infografis tidak dapat dipahami secara langsung. Akibatnya, pelaku usaha impor di Indonesia masih melakukan pencarian dan pemahaman informasi produk secara manual, yaitu membuka halaman produk satu per satu, mengunduh gambar, kemudian menerjemahkan teks Mandarin di dalamnya menggunakan layanan penerjemah gambar yang hanya mampu memproses satu gambar dalam satu waktu. Proses berulang ini membutuhkan waktu lama, terutama ketika harus menganalisis puluhan hingga ratusan produk sekaligus. Selain itu, konten pada platform 1688.com dimuat secara dinamis menggunakan teknologi AJAX dan arsitektur Single Page Application (SPA), sehingga metode pengambilan data berbasis HTTP request konvensional tidak efektif diterapkan pada platform tersebut. Kondisi ini menjadi justifikasi utama kebutuhan akan sistem yang mampu menerjemahkan teks pada gambar produk secara otomatis dan langsung di halaman 1688.com.

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.

2. KAJIAN TEORITIS

Web Scraping

Web scraping merupakan teknik untuk memperoleh data secara otomatis dari halaman web dengan mengekstraksi informasi yang terdapat pada struktur HyperText Markup Language (HTML) maupun Document Object Model (DOM). Teknik ini banyak dimanfaatkan dalam bidang *business intelligence*, analisis data, penelitian, serta pengembangan sistem informasi karena mampu mengumpulkan data dalam jumlah besar secara efisien. Menurut

Khder (2021), *web scraping* memungkinkan proses ekstraksi data dari situs web secara otomatis sehingga menghasilkan data yang lebih cepat, konsisten, dan akurat dibandingkan proses pengambilan data secara manual.

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 akurasinya dipengaruhi oleh kualitas gambar, pencahayaan, jenis font, serta kualitas pra-pemrosesan citra seperti binarization dan noise reduction (Chen et al., 2021; Gabor et al., 2018). Karakter Mandarin dikenal memiliki tingkat kesulitan pengenalan yang tinggi; Wang et al. (2020) menunjukkan bahwa kombinasi pra-pemrosesan citra dan model koreksi pasca-OCR dapat meningkatkan akurasi pengenalan karakter Tionghoa hingga sekitar 12%, sehingga optimasi tahap pra-pemrosesan menjadi target utama dalam penelitian ini.

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.

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).*

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.

Large Language Model (LLM) Gemini

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 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.

V-Model (Verification and Validation Model)

Verification and Validation Model (V-Model) merupakan model Software Development Life Cycle (SDLC) yang mengintegrasikan aktivitas verifikasi dan validasi pada setiap tahapan pengembangan, sehingga setiap tahap perancangan memiliki tahapan pengujian yang sesuai, berbeda dengan model Waterfall yang memisahkan proses pengembangan dan pengujian (OpenStax, 2024).

API Google Translate

Google Translate API merupakan layanan penerjemahan mesin (Machine Translation) berbasis Neural Machine Translation (NMT) dari Google Cloud yang memungkinkan aplikasi menerjemahkan teks ke lebih dari 100 bahasa secara otomatis melalui antarmuka REST tanpa membangun model penerjemahan sendiri.

3. METODE PENELITIAN

Penelitian ini menggunakan metode *Research and Development* (R&D) dengan model pengembangan V-Model untuk mengembangkan sistem yang mampu membantu pelaku usaha dalam memperoleh, menerjemahkan, dan mengelola informasi produk dari platform 1688.com.

Metode R&D dipilih karena penelitian tidak hanya bertujuan mengidentifikasi permasalahan, tetapi juga menghasilkan produk berupa sistem yang dapat digunakan untuk menyelesaikan permasalahan tersebut. Model V-Model digunakan karena menyediakan tahapan pengembangan yang sistematis serta menghubungkan setiap tahap pengembangan dengan proses verifikasi dan validasi sehingga kualitas sistem dapat dievaluasi secara menyeluruh.

Tahapan penelitian dimulai dengan identifikasi masalah dan studi literatur. Pada tahap ini dilakukan observasi terhadap platform 1688.com untuk mengidentifikasi kendala yang dihadapi pelaku usaha dalam memperoleh informasi produk, khususnya informasi yang masih menggunakan bahasa Mandarin dan disajikan dalam bentuk gambar. Studi literatur dilakukan dengan mengkaji berbagai referensi yang berkaitan dengan web scraping, Optical Character Recognition (OCR), Google Translate API, image text swapping, Chrome Extension, Large Language Model (LLM), dan Firefly Algorithm sebagai dasar dalam menentukan metode dan teknologi yang digunakan.

4. HASIL DAN PEMBAHASAN

Implementasi Chrome Extension

Chrome Extension diimplementasikan sebagai antarmuka utama yang digunakan pengguna untuk berinteraksi dengan sistem. Ekstensi ini dikembangkan menggunakan HTML, CSS, dan JavaScript dengan arsitektur Manifest V3. Chrome Extension berfungsi sebagai media penghubung antara pengguna dengan backend, sehingga proses scraping, pengiriman data, serta penerimaan hasil pemrosesan dapat dilakukan secara langsung pada halaman produk platform 1688.

Saat pengguna membuka halaman produk 1688, content script yang berjalan pada Chrome Extension dapat mengakses struktur Document Object Model (DOM) dari halaman tersebut. Selanjutnya data produk yang diperoleh dikirimkan ke backend melalui Application Programming Interface (API) untuk diproses lebih lanjut, seperti OCR, penerjemahan gambar, serta penyimpanan hasil scraping.

Implementasi Chrome Extension disusun secara modular dengan membagi fungsi ke dalam beberapa komponen utama. Pembagian komponen tersebut ditunjukkan pada Tabel 4.

Tabel 4. Komponen pada *Chrome Extension.*

Komponen	Fungsi
manifest.json	Berisi konfigurasi utama Chrome Extension, seperti nama ekstensi, versi, izin (*permissions*), *content script*, *background service worker*, serta halaman popup yang digunakan.
content.js	Bertugas melakukan ekstraksi data produk dari halaman 1688.com melalui DOM, menampilkan antarmuka ekstensi menggunakan Shadow DOM, serta melakukan pembaruan tampilan berdasarkan hasil pemrosesan yang diterima dari backend.
background.js	Berfungsi sebagai *service worker* yang menangani komunikasi antara Chrome Extension dengan Backend Flask melalui REST API serta mengelola proses yang berjalan di latar belakang.
popup.html	Menyediakan antarmuka popup yang ditampilkan ketika pengguna menekan ikon Chrome Extension pada toolbar browser.
icons/	Berisi ikon yang digunakan sebagai identitas Chrome Extension pada browser Google Chrome.

Berdasarkan Tabel 4, setiap komponen pada Chrome Extension memiliki tanggung jawab yang berbeda. Pembagian ini menerapkan prinsip modularitas sehingga konfigurasi ekstensi, antarmuka pengguna, proses DOM Scraping, komunikasi dengan backend, dan pengaturan sistem dapat dikembangkan serta dipelihara secara terpisah.

Implementasi Antarmuka *Chrome Extension*

Pada penelitian ini dikembangkan sebuah Chrome Extension bernama *1688 Assistant*. Antarmuka ekstensi ditampilkan secara otomatis pada halaman detail produk platform 1688 setelah pengguna mengaktifkan ekstensi pada peramban Google Chrome.

Gambar 3 menunjukkan implementasi antarmuka Chrome Extension yang terdiri atas beberapa komponen utama. Antarmuka dirancang dengan tampilan yang sederhana agar pengguna dapat menjalankan setiap fungsi sistem secara mudah. Komponen yang tersedia meliputi informasi status halaman, fitur *Import Data*, *Terjemahkan Gambar*, *Download Gambar Katalog*, akses menuju *Web Dashboard*, serta menu pengaturan bahasa penerjemahan.



Gambar 3. Implementasi Antarmuka *Chrome Extension*

Berdasarkan Gambar 3, antarmuka Chrome Extension menyediakan tombol untuk menutup panel serta informasi status halaman produk yang sedang dibuka, di samping lima fitur utama yang dijelaskan lebih lanjut pada subbab berikut, yaitu Import Data untuk

mengambil data produk melalui DOM Scraping, Terjemahkan Gambar untuk memproses OCR, penerjemahan, dan image text swapping pada gambar produk, Download Gambar Katalog untuk mengunduh seluruh gambar produk dalam format ZIP, Dashboard untuk membuka Web Dashboard, serta Pengaturan untuk memilih bahasa tujuan penerjemahan.

Berdasarkan implementasi yang telah dilakukan, Chrome Extension berperan sebagai antarmuka utama yang menghubungkan pengguna dengan seluruh layanan sistem. Chrome Extension bertugas mengambil data dari halaman produk, mengirimkan permintaan ke backend, menerima status proses, serta menampilkan kembali hasil pemrosesan kepada pengguna. Seluruh proses komputasi, seperti OCR, penerjemahan teks, Image Text Swapping, penyimpanan data, hingga generasi ringkasan produk menggunakan Google Gemini API dilakukan pada sisi backend. Dengan pembagian tugas tersebut, Chrome Extension tetap ringan karena hanya menangani interaksi pengguna, sedangkan proses komputasi yang membutuhkan sumber daya lebih besar dipusatkan pada backend.

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 1688 secara otomatis. Fitur ini membantu pengguna memahami informasi visual produk tanpa perlu melakukan proses penerjemahan secara manual maupun menggunakan aplikasi penerjemah lain.

Ketika pengguna menekan tombol Terjemahkan Gambar, Chrome Extension melakukan ekstraksi URL seluruh gambar produk dari halaman detail 1688 menggunakan teknik DOM Scraping. URL gambar beserta bahasa tujuan yang telah dipilih pada menu Pengaturan kemudian dikirimkan ke backend melalui REST API untuk diproses lebih lanjut.



Gambar 4. Implementasi dan Hasil Fitur Terjemahkan Gambar pada Chrome Extension

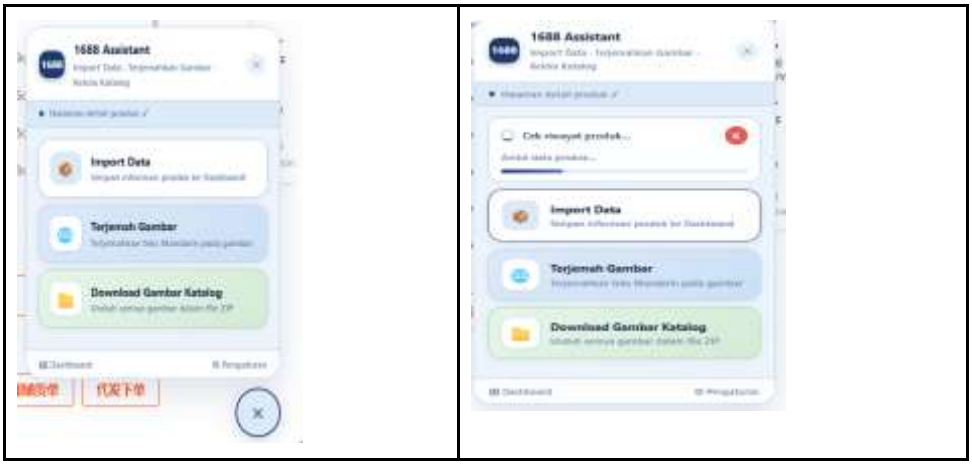
Pada sisi backend, gambar diproses melalui beberapa tahapan yang meliputi OCR untuk mendeteksi teks pada gambar, penerjemahan teks ke bahasa tujuan, serta *Image Text Swapping* untuk menggantikan teks Mandarin dengan hasil terjemahan. Setelah seluruh tahapan selesai, backend mengembalikan gambar hasil translasi kepada Chrome Extension.

Chrome Extension kemudian memperbarui tampilan halaman produk 1688 dengan menampilkan gambar hasil translasi sebagai pengganti gambar asli. Dengan pendekatan tersebut, pengguna dapat langsung melihat hasil penerjemahan pada halaman produk yang sama tanpa perlu melakukan penyegaran halaman (*refresh*) maupun berpindah ke aplikasi lain.

Berdasarkan gambar tersebut, gambar produk yang semula berisi teks berbahasa Mandarin berhasil ditampilkan kembali dengan teks hasil terjemahan sesuai bahasa yang dipilih pengguna. Perubahan dilakukan secara langsung pada halaman produk sehingga informasi visual dapat dipahami tanpa mengubah struktur maupun tata letak gambar asli.

Implementasi Fitur Import Data

Fitur Import Data diimplementasikan untuk memudahkan pengguna mengambil informasi produk dari halaman detail produk 1688 dan menyimpannya ke dalam sistem, sehingga mengurangi kebutuhan menyalin informasi produk secara manual. Saat pengguna menekan tombol Import Data, Chrome Extension mengekstraksi informasi produk—meliputi nama produk, harga, nama pemasok, spesifikasi, URL produk, gambar deskripsi, dan gambar varian—melalui struktur DOM, mengemasnya dalam format JSON, lalu mengirimkannya ke backend melalui REST API menggunakan metode HTTP POST.



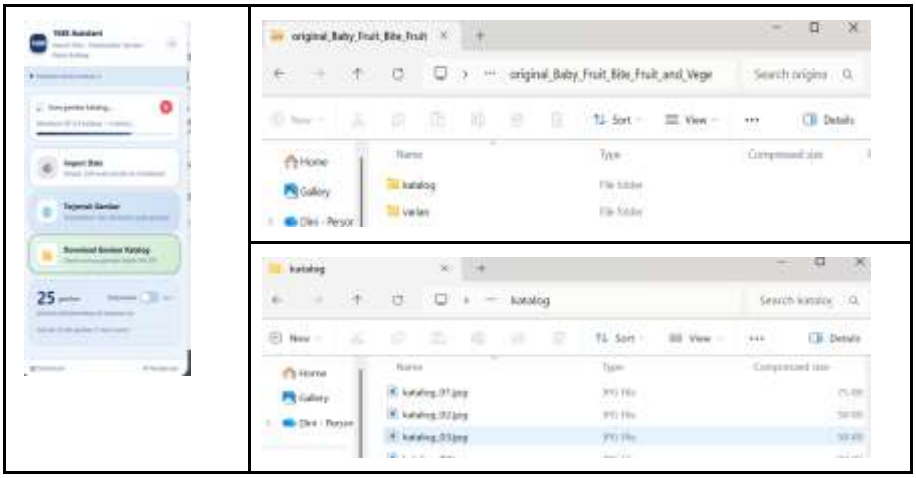
Gambar 5. Implementasi Fitur Import Data pada Chrome Extension

Setelah diterima, backend memvalidasi data sebelum menyimpannya ke basis data, dan data yang tersimpan kemudian ditampilkan pada Web Dashboard sehingga pengguna dapat mengelola informasi produk, melakukan pencarian, serta menggunakannya pada proses penerjemahan gambar maupun penyusunan katalog. Dengan demikian, fitur Import Data

membuat proses pengumpulan informasi produk lebih efisien dibandingkan pencatatan manual.

Implementasi Fitur Download Gambar Katalog

Fitur Download Gambar Katalog diimplementasikan untuk memudahkan pengguna mengunduh seluruh gambar produk pada halaman detail produk 1688 dalam satu proses, menghilangkan kebutuhan mengunduh gambar satu per satu secara manual. Ketika tombol ditekan, Chrome Extension mengidentifikasi seluruh gambar pada halaman produk dan mengelompokkannya secara otomatis menjadi gambar deskripsi produk (product description images) dan gambar varian produk (product variant images) berdasarkan struktur halaman.



Gambar 6. Implementasi Fitur Download Gambar Katalog

Seluruh gambar katalog yang telah diidentifikasi kemudian diunduh dan dikompresi ke dalam satu berkas ZIP dengan struktur folder yang memisahkan gambar deskripsi dan gambar varian, sehingga hasil unduhan lebih terorganisir dan mempermudah pengguna dalam penyusunan katalog maupun proses pemasaran.

Implementasi Pengaturan Bahasa

Fitur Pengaturan Bahasa diimplementasikan untuk memberikan fleksibilitas kepada pengguna dalam menentukan bahasa tujuan penerjemahan gambar. Bahasa yang dipilih pada menu Pengaturan disimpan sebagai konfigurasi Chrome Extension sehingga digunakan secara otomatis pada setiap proses Terjemahkan Gambar berikutnya tanpa perlu dipilih ulang.

Saat fitur Terjemahkan Gambar dijalankan, Chrome Extension mengirimkan konfigurasi bahasa tersimpan bersama URL gambar ke backend melalui REST API sebagai parameter bahasa tujuan penerjemahan. Dengan mekanisme ini, sistem dapat mendukung penerjemahan ke beberapa bahasa tanpa memerlukan perubahan pada proses OCR maupun Image Text Swapping.

Gambar original	Hasil terjemahan Bahasa Indonesia	Hasil terjemahan Bahasa Inggris
		

Gambar 7. Hasil Implementasi Pengaturan Bahasa

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 dilakukan menggunakan bahasa pemrograman Python dengan framework Flask sebagai pusat koordinasi antar modul. Flask berperan dalam mengelola komunikasi antara Chrome Extension, layanan pemrosesan data, serta basis data melalui REST API.

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.

Implementasi Modul OCR

Modul Optical Character Recognition (OCR) digunakan untuk mendeteksi dan mengekstraksi teks pada gambar produk 1688.com menggunakan library EasyOCR yang mendukung karakter Mandarin Simplified dan bahasa Inggris. Sebelum proses OCR, gambar melalui tahap image preprocessing berupa penyesuaian ukuran dan penerapan Contrast Limited Adaptive Histogram Equalization (CLAHE) secara adaptif pada citra berkontras rendah untuk meningkatkan keterbacaan teks tanpa menambah waktu pemrosesan pada gambar yang sudah berkualitas baik.

Setelah tahap preprocessing selesai dilakukan, gambar diproses menggunakan EasyOCR untuk mendeteksi posisi teks (bounding box) sekaligus mengenali karakter pada

gambar. Hasil OCR berupa teks beserta lokasi kemunculannya digunakan sebagai dasar dalam proses translasi dan penggantian teks pada gambar.




```
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)
------------------------------------------------------------
```

Gambar 8. Hasil Deteksi Bounding Box Menggunakan EasyOCR

Berdasarkan hasil pengujian, teks berukuran kecil memiliki akurasi lebih rendah, sehingga sistem menerapkan mekanisme Retry OCR dengan memotong ulang area teks tersebut dan menggunakan hasil dengan nilai confidence lebih tinggi sebagai keluaran akhir.



Gambar 9. Mekanisme Retry OCR

Sebelum diteruskan ke modul translasi, hasil OCR melalui tahap Post Processing untuk menormalisasi teks dan memperbaiki kesalahan karakter umum (misalnya huruf vs angka pada spesifikasi produk) guna meningkatkan akurasi translasi.

Implementasi Modul Translasi

Modul translasi menerjemahkan hasil ekstraksi teks dari modul OCR ke bahasa tujuan yang dipilih pengguna menggunakan layanan Google Translate, sebelum diteruskan ke proses Image Text Swapping. Setiap teks hasil OCR diterjemahkan secara individual agar informasi posisi (bounding box) dan urutan teks tetap terjaga, sehingga hasil translasi dapat digunakan kembali pada proses penggantian teks tanpa mengubah tata letak maupun struktur gambar produk.

Tabel 5. Hasil Implementasi Modul Translasi

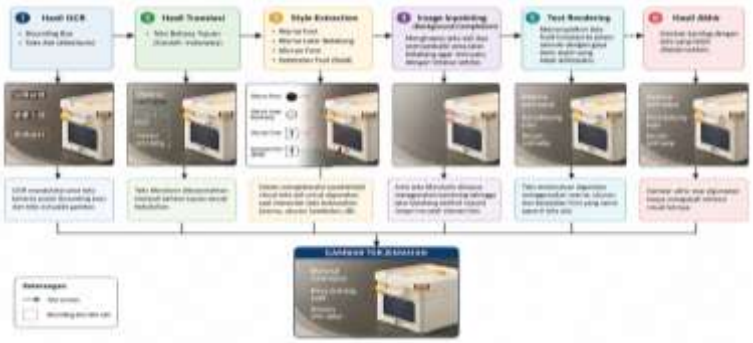
	Ditemukan 7 region teks (7 Mandarin, 0 non-Mandarin) Ditemukan 7 teks ========== GOOGLE TRANSLATE DEBUG OCR : 低功耗强续航 Translated : Konsumsi daya rendah dan masa pakai baterai yang lama ==========GOOGLE TRANSLATE DEBUG OCR : 7天畅快用 Translated : Nikmati selama 7 hari ==========GOOGLE TRANSLATE DEBUG OCR : 蓝牙低功耗芯片。耗能低, Translated : Chip Bluetooth hemat energi. Konsumsi energi rendah, ========== GOOGLE TRANSLATE DEBUG OCR : 性能稳定; Translated : Kinerja yang stabil; ========== GOOGLE TRANSLATE DEBUG OCR : 一次充电 Translated : Satu biaya ==========GOOGLE TRANSLATE DEBUG 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 DEBUG 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 tanpa mengubah urutan hasil OCR. Hasil translasi tersebut selanjutnya digunakan sebagai masukan pada proses Image Text Swapping

untuk menggantikan teks Mandarin pada gambar produk dengan hasil terjemahan sesuai bahasa yang dipilih pengguna.

Implementasi Image Text Swapping

Modul Image Text Swapping bertugas mengganti teks berbahasa Mandarin pada gambar produk menjadi hasil terjemahan melalui tiga tahap utama, yaitu Style Extraction, Image Inpainting, dan Text Rendering, sebagaimana ditunjukkan pada Gambar 10.



Gambar 10. Alur Implementasi Image *Text Swapping*

Berdasarkan Gambar 10, proses Image Text Swapping diawali dengan menerima hasil OCR berupa posisi teks (bounding box) dan isi teks yang telah berhasil diekstraksi dari gambar produk. Selanjutnya, sistem memperoleh hasil terjemahan dari modul translasi yang akan digunakan sebagai pengganti teks berbahasa Mandarin pada gambar.

Tahap berikutnya adalah Style Extraction, yaitu proses untuk mengekstraksi karakteristik visual dari area teks asli. Modul ini diimplementasikan menggunakan OpenCV dengan memanfaatkan koordinat *bounding box* hasil OCR sebagai acuan untuk mengambil area teks pada gambar. Dari area tersebut, sistem menganalisis beberapa karakteristik visual, seperti warna dominan teks, warna latar belakang, ukuran huruf, ketebalan tulisan, serta orientasi teks. Informasi tersebut kemudian disimpan sebagai referensi untuk proses penempatan teks hasil terjemahan agar tetap selaras dengan tampilan gambar asli.

Setelah karakteristik visual diperoleh, sistem melakukan proses Image Inpainting untuk menghapus teks Mandarin pada gambar. Tahap ini diimplementasikan menggunakan metode Telea Inpainting yang tersedia pada pustaka OpenCV. Proses inpainting memanfaatkan *mask* yang dibentuk berdasarkan koordinat *bounding box* sehingga area yang sebelumnya berisi teks direkonstruksi menggunakan informasi piksel di sekitarnya. Hasil dari proses ini berupa latar belakang yang telah dibersihkan dan siap digunakan untuk penempatan teks baru.

Tahap selanjutnya adalah Text Rendering, yaitu proses menggambar hasil terjemahan pada posisi yang sama dengan teks asli berdasarkan informasi *bounding box* hasil OCR. Modul ini diimplementasikan menggunakan pustaka Pillow (PIL) untuk melakukan rendering teks

karena mendukung penggunaan berbagai jenis font serta menghasilkan kualitas teks yang lebih baik dibandingkan fungsi penulisan teks bawaan OpenCV. Pada tahap ini sistem menyesuaikan ukuran huruf, warna teks, ketebalan tulisan, serta posisi penempatan berdasarkan karakteristik visual yang diperoleh dari proses *Style Extraction*. Penyesuaian tersebut bertujuan menghasilkan tampilan teks yang tetap selaras dengan desain gambar produk, meskipun pada beberapa gambar dengan latar belakang yang kompleks hasil visual belum sepenuhnya identik dengan teks asli.

Tabel 6. Hasil Implementasi Modul Image Text Swapping

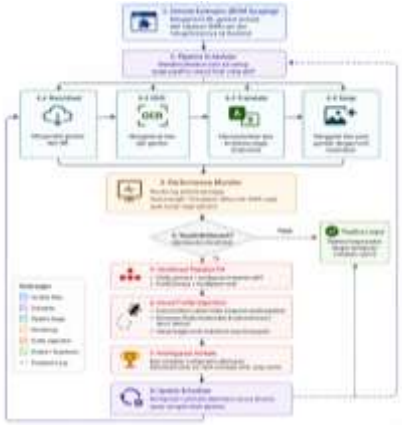
Sebelum	**Sesudah**
	
	

Berdasarkan hasil implementasi, modul Image Text Swapping mampu menghasilkan gambar katalog dengan teks berbahasa Indonesia yang tetap mempertahankan tata letak dan karakteristik visual gambar asli. Dengan demikian, informasi produk dapat dipahami oleh pengguna.

Implementasi Firefly Algorithm

Firefly Algorithm diimplementasikan sebagai mekanisme optimasi adaptif untuk menentukan konfigurasi Pipeline Scheduler pada proses pengolahan gambar produk, bukan untuk menentukan urutan pemrosesan gambar maupun scraping. Berbeda dengan scheduler statis, konfigurasi pada penelitian ini dapat diperbarui secara dinamis sesuai kondisi aktual pipeline.

Setelah Chrome Extension memperoleh URL gambar melalui DOM Scraping, backend mendistribusikan setiap gambar ke empat tahapan pemrosesan, yaitu Download, OCR, Translate, dan Image Text Swapping, yang masing-masing dikendalikan oleh Stage Controller sesuai konfigurasi scheduler aktif, sebagaimana ditunjukkan pada Gambar 11.



Gambar 11. Diagram Implementasi *Firefly Algorithm*

Mekanisme optimasi dimulai ketika Performance Monitor mendeteksi ketidakseimbangan beban kerja (bottleneck) berdasarkan kondisi queue, active task, throughput, dan jumlah retry pada setiap tahapan. Ketika kriteria tertentu terpenuhi, sistem mengambil snapshot kondisi pipeline sebagai dasar evaluasi, lalu membangkitkan populasi awal (initial population) konfigurasi Pipeline Scheduler—firefly pertama menggunakan konfigurasi aktif, sedangkan firefly lainnya dibangkitkan secara acak. Setiap kandidat dievaluasi menggunakan fungsi fitness untuk menentukan tingkat kecerahan (brightness), sehingga firefly dengan fitness lebih tinggi menjadi tujuan perpindahan bagi firefly lainnya hingga mencapai iterasi maksimum atau tidak terjadi peningkatan fitness selama beberapa iterasi (patience). Pipeline tetap memproses tugas yang sedang berjalan menggunakan konfigurasi sebelumnya sehingga aktivitas sistem tidak terhenti selama optimasi berlangsung.

Setiap firefly merepresentasikan satu kombinasi konfigurasi *Pipeline Scheduler* yang terdiri atas empat parameter, yaitu *Download Limit*, *OCR Limit*, *Translate Limit*, dan *Swap Limit*. Keempat parameter tersebut menentukan jumlah maksimum tugas yang dapat diproses secara bersamaan pada masing-masing tahapan pipeline. Dengan demikian, proses optimasi yang dilakukan Firefly Algorithm tidak mengubah urutan eksekusi tugas, tetapi mencari kombinasi konfigurasi yang paling sesuai dengan kondisi beban kerja pipeline sehingga kapasitas pemrosesan pada setiap tahapan dapat menjadi lebih seimbang.

Setelah optimasi selesai, Firefly Algorithm memilih konfigurasi dengan nilai fitness terbaik untuk memperbarui SchedulerConfig, sehingga setiap Stage Controller memperbarui batas konkurensi pada tahap Download, OCR, Translate, dan Image Text Swapping secara

dinamis tanpa menghentikan pipeline yang sedang berjalan—tugas yang telah memperoleh slot tetap menggunakan konfigurasi sebelumnya, sedangkan tugas berikutnya menggunakan konfigurasi hasil optimasi. Dengan mekanisme ini, Firefly Algorithm berperan sebagai optimasi adaptif yang mengurangi potensi bottleneck dan meningkatkan keseimbangan beban kerja antar tahap.

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.

Web Dashboard menyediakan tiga menu utama pendukung: Menu Data Produk yang menampilkan seluruh data produk hasil scraping—lengkap dengan fitur pencarian dan tombol Detail—sebagai pusat pengelolaan data; Menu Katalog Produk untuk mengelola hasil penerjemahan sebelum disusun menjadi katalog, dilengkapi fitur Generate Deskripsi berbasis Google Gemini AI serta Download Gambar Original maupun Gambar Terjemahan; dan Menu Gambar Terjemahan yang menampilkan seluruh gambar hasil Image Text Swapping secara berpasangan (sebelum-sesudah) sehingga pengguna dapat memeriksa kualitas dan tata letak hasil penerjemahan.

Pengujian Sistem

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, Firefly Algorithm, serta Web Dashboard. 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.

Hubungan antara tahapan Perancangan dan tahapan pengujian berdasarkan metode V-Model ditunjukkan pada Gambar 12.



Gambar 12. Hubungan Perancangan dan Pengujian Berdasarkan V-Model

Tabel 7. Ringkasan Hasil Unit Testing, Integration Testing, dan System Testing

Unit Testing	Seluruh modul utama—mulai dari pengambilan data hingga penyimpanan ke basis data—berhasil dijalankan secara mandiri dan menghasilkan keluaran sesuai masukan yang diberikan.
Integration Testing	Seluruh modul berhasil berinteraksi dan bertukar data sesuai alur yang dirancang, mulai dari DOM scraping, OCR, penerjemahan, image text swapping, optimasi Firefly Algorithm, hingga penyimpanan dan penyajian data, tanpa ditemukan kesalahan integrasi.
System Testing – Fungsional	Seluruh fungsi utama sistem (scraping, OCR, penerjemahan, image text swapping, optimasi Pipeline Scheduler, penyimpanan data, penyajian melalui Web Dashboard) berjalan sesuai kebutuhan yang ditentukan.
System Testing – Non-fungsional	Sistem memenuhi aspek kompatibilitas, kemudahan penggunaan, integritas data, dan konektivitas layanan eksternal: berjalan baik pada Google Chrome, menjaga konsistensi data, serta berhasil terintegrasi dengan Google Translate API dan Gemini API.

Pengujian *Firefly Algorithm*

Pengujian Firefly Algorithm dilakukan untuk mengevaluasi kemampuan algoritma dalam mengoptimalkan Pipeline Scheduler. Sebelum diterapkan, sistem memproses gambar secara sekuensial (sequential processing)—setiap gambar diproses bergantian dari DOM, OCR, penerjemahan, hingga image text swapping sebelum gambar berikutnya diproses—sehingga sumber daya komputasi belum dimanfaatkan secara optimal. Penelitian ini kemudian mengimplementasikan Pipeline Scheduler yang dioptimalkan Firefly Algorithm agar beberapa gambar dapat diproses bersamaan pada tahapan berbeda, dengan konfigurasi scheduler ditentukan secara adaptif berdasarkan kondisi pipeline yang sedang berjalan.

Ilustrasi perbedaan mekanisme pemrosesan sebelum dan sesudah penerapan Firefly Algorithm ditunjukkan pada Gambar 13.



Gambar 13. Perbandingan Sebelum (Sequential Processing) dan Sesudah (Pipeline Scheduler) Penerapan Firefly Algorithm

Pengujian dilakukan dengan mengamati proses optimasi Firefly Algorithm terhadap konfigurasi Pipeline Scheduler, yang dijalankan ketika Performance Monitor mendeteksi kondisi tertentu seperti bottleneck, penurunan throughput, peningkatan retry count, atau penggunaan memori berlebih, dengan snapshot kondisi pipeline sebagai masukan proses optimasi.

Parameter yang dioptimalkan oleh Firefly Algorithm ditunjukkan pada Tabel 8.

Tabel 8. Parameter Optimasi *Firefly Algorithm*

Parameter	**Keterangan**
Max Active Tasks	Jumlah maksimum tugas yang dapat diproses secara bersamaan.
Release Interval	Interval pelepasan tugas baru ke dalam pipeline.
Priority Weight	Bobot prioritas antar tahapan pipeline.

Selama proses optimasi, Firefly Algorithm menghasilkan konfigurasi *Pipeline Scheduler* yang berbeda sesuai dengan kondisi pipeline yang terdeteksi oleh *Performance Monitor*. Konfigurasi yang dihasilkan berupa penyesuaian jumlah *worker* pada setiap tahapan pipeline, yaitu *Download*, *OCR*, *Translate*, dan *Image Text Swapping*. Hasil optimasi tersebut kemudian dicatat pada tabel fa_optimization_log sebagai bagian dari proses evaluasi sistem. Contoh hasil optimasi ditunjukkan pada Tabel 9.

Tabel 9. Hasil Optimasi Konfigurasi Pipeline Scheduler Menggunakan Firefly Algorithm

Trigger	Konfigurasi Sebelum Optimasi (D,O,T,S)	Konfigurasi Setelah Optimasi (D,O,T,S)	Best Fitness	Durasi Optimasi (ms)
Bottleneck Download	(5,1,3,1)	(5,1,2,1)	0.753733	12
Bottleneck OCR	(5,1,2,1)	(3,2,1,2)	0.725255	5
RAM High	(5,1,1,1)	(5,1,1,1)	0.876033	9

Keterangan:

- **D** = Download Worker
- = OCR Worker
- **T** = Translate Worker
- **S** = Swap Worker

Berdasarkan Tabel 9, Firefly Algorithm menghasilkan konfigurasi berbeda sesuai kondisi (trigger) yang terdeteksi: pada bottleneck download, alokasi worker pada tahap terkait ditingkatkan; pada bottleneck OCR, beban kerja didistribusikan lebih seimbang antar tahapan; sedangkan pada kondisi RAM High, konfigurasi dipertahankan karena telah optimal.

Hasil pengujian tersebut menunjukkan bahwa Firefly Algorithm tidak mengubah urutan proses pengolahan gambar, melainkan mengoptimalkan konfigurasi *Pipeline Scheduler* melalui penyesuaian jumlah *worker* pada setiap tahapan pipeline. Dengan mekanisme tersebut, beberapa gambar dapat diproses secara bersamaan pada tahapan yang berbeda sehingga pemanfaatan sumber daya sistem menjadi lebih efisien dibandingkan mekanisme *sequential processing*.

User Acceptance Testing (UAT)

User Acceptance Testing (UAT) dilakukan untuk mengevaluasi tingkat penerimaan pengguna terhadap sistem berdasarkan kebutuhan proses bisnis nyata, berbeda dari pengujian fungsional yang berfokus pada kinerja sistem semata.

Pada penelitian ini, responden dipilih menggunakan teknik purposive sampling, yaitu teknik pemilihan responden berdasarkan kesesuaian peran dan pengalaman terhadap sistem yang dikembangkan. Pengujian melibatkan tiga responden yang terdiri atas satu Owner bisnis impor, satu Staf Purchasing, dan satu Admin Marketplace yang sehari-hari bertugas menyusun katalog produk. Ketiga responden dipilih karena mewakili pengguna utama (*key users*) yang secara langsung terlibat dalam proses pencarian produk, pengelolaan informasi produk, serta penyusunan katalog dari platform 1688.

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

Berdasarkan hasil User Acceptance Testing, sistem memperoleh tingkat penerimaan pengguna sebesar 84.16% 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.

5. KESIMPULAN DAN SARAN

Kesimpulan

Penelitian ini berhasil mengembangkan sistem berbasis Chrome Extension yang mampu melakukan pengambilan informasi produk dari platform 1688.com secara otomatis melalui teknik DOM scraping, serta mengintegrasikan proses Optical Character Recognition (OCR), Google Translate API, image text swapping, dan penyusunan deskripsi produk dalam satu sistem. Hasil pengujian menunjukkan bahwa seluruh fungsi utama sistem berjalan sesuai dengan kebutuhan yang telah ditetapkan.

Penerapan Firefly Algorithm pada Pipeline Scheduler terbukti mampu mengoptimalkan konfigurasi proses scraping dan pengolahan gambar secara adaptif sesuai kondisi sistem sehingga meningkatkan efisiensi pemrosesan. Selain itu, hasil pengujian menggunakan pendekatan V-Model yang meliputi Unit Testing, Integration Testing, System Testing, dan User Acceptance Testing (UAT) menunjukkan bahwa sistem memiliki tingkat penerimaan pengguna sebesar 84,16% dengan kategori Sangat Baik. Dengan demikian, sistem yang dikembangkan dinilai layak digunakan untuk membantu pelaku usaha memperoleh dan mengelola informasi produk dari platform 1688.com secara lebih cepat, akurat, dan efisien.

Saran

Pengembangan selanjutnya disarankan untuk membandingkan kinerja Firefly Algorithm dengan algoritma optimasi lainnya serta melakukan pengujian kuantitatif terhadap peningkatan performa sistem pada skala data yang lebih besar. Selain itu, peningkatan akurasi OCR, kualitas image text swapping, dukungan terhadap lebih banyak bahasa, serta penerapan mekanisme keamanan, caching, dan pemrosesan batch perlu dipertimbangkan agar sistem

memiliki kinerja yang lebih optimal. Pengguna juga disarankan tetap melakukan verifikasi terhadap hasil terjemahan sebelum digunakan sebagai informasi produk untuk memastikan kesesuaian konteks.

DAFTAR REFERENSI

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

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

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

Google Chrome for Developers. (2024). *Manifest V3 Overview*. Google Developer Documentation. https://developer.chrome.com/docs/extensions/mv3/

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. 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

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