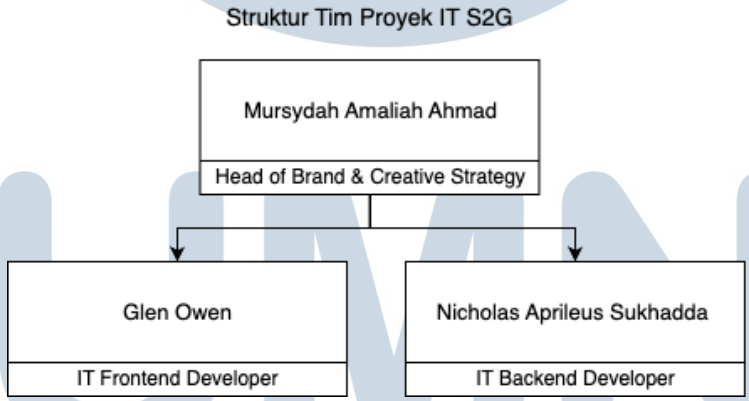
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan magang di PT Sinar Sukacita Gemilang, posisi ditempatkan pada divisi *Information Technology* (IT) di bawah bimbingan Ibu Mursydah Amaliah Ahmad selaku *Head of Brand & Creative Strategy*. Yang mempunyai peran dalam merancang ide pengembangan *website Shine2Gether*, termasuk menentukan konsep fitur, alur interaksi pengguna, serta desain visual *website*.
Kegiatan pengembangan sistem dilakukan secara kolaboratif oleh dua peserta magang, yang berperan sebagai *Frontend Developer* dan sebagai *Backend Developer*. Pada laporan Magang ini mengulas bagian *backend* seperti pengelolaan data, integrasi fitur *AI*, serta pengembangan *API* agar seluruh fungsi dapat berjalan terhubung dengan baik. Struktur koordinasi selama magang dapat dilihat pada Gambar 3.1.



Gambar 3.1. Struktur Koordinasi Divisi IT PT Sinar Sukacita Gemilang

Koordinasi tim dilakukan secara rutin melalui pertemuan mingguan bersama pembimbing lapangan untuk membahas perkembangan fitur dan desain sistem. Komunikasi harian dilakukan menggunakan *Google Meet* dan *WhatsApp Group*, Dengan sistem kerja ini, setiap anggota dapat berkolaborasi secara efektif dan memastikan proses pengembangan *website Shine2Gether* berjalan sesuai arah yang telah direncanakan.

3.2 Tugas yang Dilakukan

Selama kegiatan magang di PT Sinar Sukacita Gemilang, kegiatan difokuskan pada pengembangan sistem *website Shine2Gether*. Tugas utama berfokus pada aspek teknis yang melibatkan proses pengembangan, integrasi, dan pengujian sistem. Rangkuman *Tools* dan deskripsi tugas selama pengembangan ditunjukkan pada Tabel 3.1 berikut.

Tabel 3.1. Rangkuman *Tools* yang digunakan untuk pengembangan *website* SHiNE2GeTHER

Teknologi / *Tools*	**Deskripsi Penggunaan**
Visual Studio Code	*Code editor* utama untuk menulis, men-*debug*, dan mengelola seluruh kode sumber proyek *frontend* maupun *backend*.
Command Prompt / Terminal	Menjalankan perintah *server*, *build process*, manajemen kontainer *Docker*, serta instalasi paket dependensi.
TypeScript	Bahasa pemrograman utama sisi klien untuk memastikan kode tipe-aman (*type-safe*), terstruktur, dan meminimalisir kesalahan.
React + Vite	Membangun antarmuka pengguna (*UI*) yang interaktif. *Vite* digunakan untuk mempercepat proses kompilasi dan *hot-reload*.
TailwindCSS	*Framework CSS* berbasis *utility-first* untuk mempercepat proses *styling* tampilan agar konsisten di berbagai ukuran layar.
Python (Flask)	Mengembangkan arsitektur *backend (API)*, menangani logika bisnis sistem, dan memproses data sebelum disimpan ke basis data.

Tabel 3.1. Rangkuman *Tools* yang digunakan untuk pengembangan *website* SHiNE2GeTHER (Lanjutan)

Teknologi / *Tools*	**Deskripsi Penggunaan**
MySQL	Sistem manajemen basis data relasional untuk menyimpan data pengguna, riwayat analisis kulit, dan produk.
OpenAI API	Mesin kecerdasan buatan (GPT-5o-mini) yang diintegrasikan untuk menganalisis kondisi kulit dan memberikan rekomendasi produk.
face-api.js	Pustaka sisi klien untuk mendeteksi wajah, pencahayaan, dan posisi kepala secara *real-time* sebelum gambar diproses.
Docker	Platform kontainerisasi untuk membungkus aplikasi dan dependensinya agar berjalan konsisten di berbagai lingkungan.
Git & GitHub	Sistem kontrol versi (*Version Control System*) untuk manajemen kode, pelacakan perubahan, dan kolaborasi tim.
Google OAuth & reCAPTCHA	Layanan pihak ketiga untuk mempermudah proses autentikasi (Login Google) dan mengamankan sistem dari akses bot.

Seluruh aspek teknis tersebut digunakan secara terintegrasi untuk memastikan sistem *website Shine2Gether* berjalan dengan stabil, aman, dan efisien dalam mendukung kebutuhan pengguna.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kegiatan magang berlangsung selama 14 minggu dengan fokus utama pada pengembangan sistem *backend* untuk website Shine2Gether. Kegiatan dimulai dengan tahap orientasi dan pengenalan lingkungan kerja, dilanjutkan dengan pengembangan fitur inti seperti autentikasi, integrasi *API* kecerdasan buatan (*OpenAI*), hingga manajemen infrastruktur menggunakan *Docker*. Selain fokus pada pengembangan web utama, dilakukan juga pengembangan proyek lain, yaitu

S2G Stock Management, yang berfokus pada sistem manajemen stok berbasis *Laravel* dan *Filament*. Pelaksanaan kerja magang secara rinci diuraikan pada Tabel 3.2.

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke-	**Pekerjaan yang dilakukan**
1	Masa orientasi lingkungan kerja, pemahaman aturan perusahaan, serta pengenalan dengan tim inti *Shine2Gether.*
2	Melakukan *Initial Commit* pada repositori *GitHub* dan pengembangan fitur *face tracker*.
3	Inisialisasi halaman Admin utama dan implementasi fitur Autentikasi pengguna (*Login* dan *Signup*).
4	Mengintegrasikan *OpenAI API* untuk keperluan fitur *Skin Analysis*.
5	Membuat tampilan dan fungsi *editor* artikel untuk pengguna menggunakan *library* Jodit.js.
6	Membuat fitur *Calendar Event* untuk member *BeautyEnthusiast* beserta konfigurasinya di *Admin Page.*
7	Melakukan *Bug Fixing* pada sistem Kalender, *Skin Analysis*, dan modul Artikel.
8	Mengintegrasikan *Docker* untuk keperluan *environment* sistem dan melakukan *brainstorming* untuk proyek *Stock Management.*
9	*(S2G Stock Management)* Membuat *environment* awal proyek menggunakan *Laravel, Docker*, dan *Filament,* serta fitur *Login* dan *Signup*.
10	*(S2G Stock Management)* Mengembangkan tampilan dan logika utama untuk modul Stok Barang dan pencatatan tanggal kedaluwarsa (*Expire Date*).

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

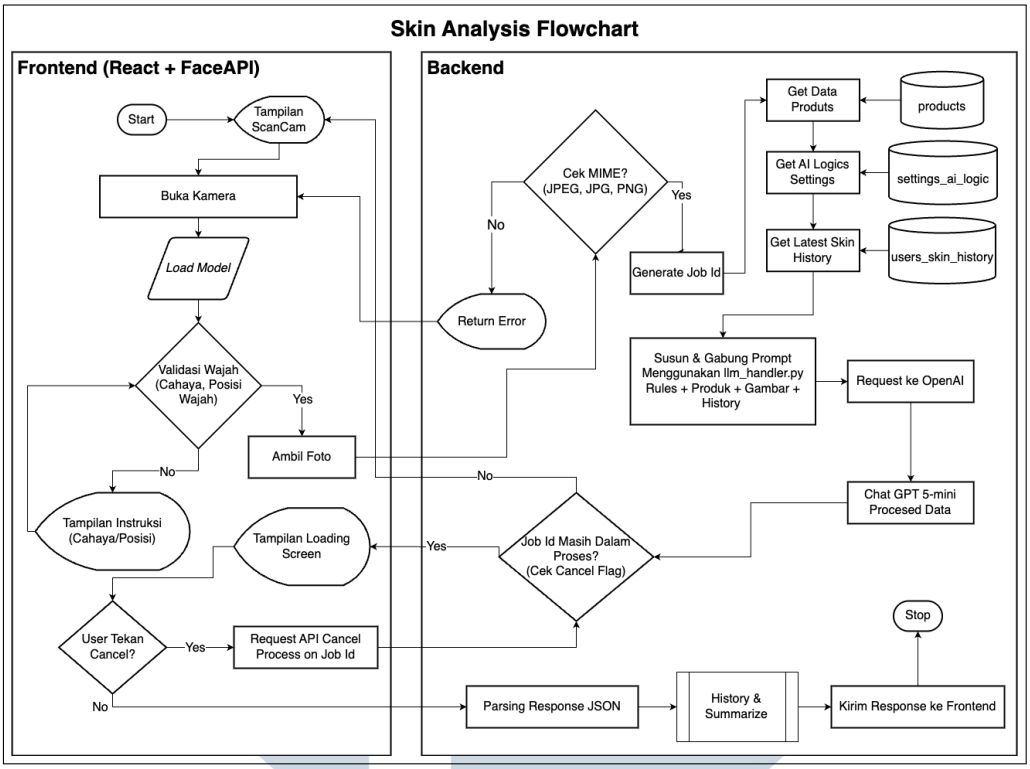
Minggu Ke-	Pekerjaan yang dilakukan
11	*(S2G Stock Management)* Mengembangkan tampilan dan logika utama untuk modul Pemesanan Barang (*Order*).
12	*(S2G Stock Management)* Membuat tampilan dan logika utama untuk modul Penerimaan Barang (*Receiving*).
13	Membuat tampilan *Frontend* untuk memvisualisasikan data *JSON* hasil jawaban dari *OpenAI Skin Analyst*.
14	Membuat sistem Gamifikasi (*Season, Challenge, Rank*) beserta konfigurasi pengaturannya di *Admin Page.*

3.3.1 Implementasi Fitur Analisis Kulit (Skin Analysis)

Fitur analisis kulit (*Skin Analysis*) merupakan fitur utama pada *website Shine2Gether* yang dikembangkan menggunakan layanan *OpenAI* (*ChatGPT*) sebagai *tools* analisisnya. Sistem ini dirancang dengan menggabungkan beberapa komponen teknis agar berjalan optimal. Implementasi fitur ini dibagi menjadi dua bagian utama, yaitu sisi pengguna (*User Side*) dan sisi administrator (*Admin Side*).

A Implementasi Sisi Pengguna (User Side)

Pada sisi pengguna, sistem menggunakan *library face-api* untuk memvalidasi kualitas gambar wajah sebelum diproses. Sedangkan pada sisi *server*, terdapat modul *llm_handler* yang bertugas menyusun instruksi *(prompt)* secara dinamis. Modul ini mengambil logika analisis *AI* serta data produk (*SKU*) dari Tabel 3.3 (yang sudah diisi datanya di halaman admin), lalu menggabungkannya dan dikirim ke *OpenAI*. Hal ini dilakukan supaya hasil analisis yang keluar tidak hanya berupa diagnosa kulit, tetapi juga rekomendasi produk yang sesuai dengan *SKU* barang yang tersedia di *database*. Alur kerja *Skin Analysis* secara keseluruhan, mulai dari pengambilan gambar hingga hasil analisis, digambarkan dalam Gambar 3.2.



Gambar 3.2. *Flowchart Skin Analysis*

Berdasarkan Gambar 3.2, mekanisme kerja sistem pada sisi pengguna dapat dijelaskan sebagai berikut:

A.1 Proses Mengambil Gambar (Frontend: Validasi Wajah Pengguna)

Proses dimulai pada sisi pengguna saat pengguna membuka fitur kamera (Pengambilan gambar). Sebelum gambar dikirim ke *server*, sistem melakukan validasi kualitas gambar secara *real-time* menggunakan *library face-api.js*. Validasi ini bertujuan untuk memastikan bahwa gambar yang diambil memiliki pencahayaan yang cukup, sudut wajah menghadap kamera, dan hanya memuat satu wajah. Implementasi logika validasi pengambilan gambar tersebut terdapat pada *file ScanCamPage.tsx* seperti ditunjukkan pada Kode 3.1 berikut.

```
// Pengecekan kondisi wajah secara real-time
useEffect(() => {
  if (detections.length === 1) {
    const face = detections[0];

    // Validasi pencahayaan (Score > 0.85 dianggap cukup terang)
    const lightingCheck = face.score > 0.85;
```

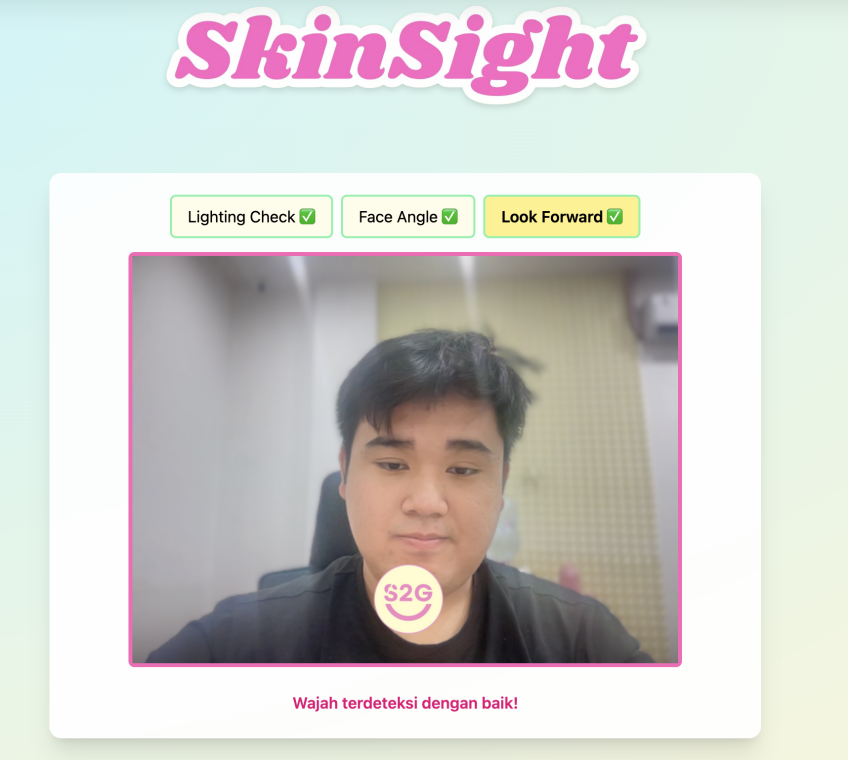
```
    setIsLightingOk(lightingCheck);

    // Validasi sudut wajah (Menghadap lurus ke depan)
    const angleCheck = Math.abs(face.angle.y) < 0.2;
    setIsAngleOk(angleCheck);

    // Update state feedback ke pengguna
    if (!lightingCheck) setFaceFeedback("Cahaya kurang terang");
    else if (!angleCheck) setFaceFeedback("Wajah harus lurus");
    else setFaceFeedback("Tahan posisi...");
  }
}, [detections]);
```

Kode 3.1: Logika validasi wajah pada sisi Frontend

Pengguna hanya dapat menekan tombol pengambilan gambar jika seluruh indikator validasi bernilai *true*. Tampilan antarmuka saat validasi berlangsung dapat dilihat pada Gambar 3.3 berikut.



Gambar 3.3. *Frontend* pengambilan gambar dengan indikator validasi

A.2 Proses Orkestrasi (Backend: Job Handling)

Setelah gambar terkirim, sisi *backend* akan menerima permintaan tersebut. Langkah pertama yang dilakukan adalah pemeriksaan *file* melalui validasi tipe *MIME*, yang kemudian diikuti dengan penyusunan konteks (*context setup*) untuk keperluan pemrosesan *AI*. Pada tahap ini, sistem juga menerapkan mekanisme keamanan untuk menolak *file* yang bukan gambar, seperti ditunjukkan pada Kode 3.2 berikut.

```
@openai_endpoints.route("/api/analyze-skin-ai", methods=["POST"])
def analyze_skin_ai():
    # 1. Validasi User Login
    user = _get_current_user()
    if not user:
        return jsonify({"error": "unauthorized"}), 401

    # 2. Validasi Format File (Security Check)
    file = request.files.get("photo")
    img_bytes = file.read()
    mime = detect_image_mime(img_bytes)

    if mime not in ["image/jpeg", "image/png", "image/webp"]:
        return jsonify({"error": "invalid_image_format"}), 400

    # 3. Lanjut ke proses AI...
```

Kode 3.2: Validasi *MIME Type* dan Inisiasi Proses di *Backend*

A.3 Proses Artificial Intelligence (Prompt Engineering dan RAG)

Proses ini adalah inti dari kecerdasan sistem. Alih-alih menggunakan *prompt* statis, pendekatan ini menyusun instruksi dinamis dengan teknik *Retrieval-Augmented Generation* (*RAG*), yang menggabungkan pengambilan data *real-time* dari basis data dengan generasi respons dari model *AI*.

A.3.1 Penjelasan Teknik Retrieval-Augmented Generation (RAG)

RAG adalah pendekatan yang menyelaraskan tiga tahap kritis:

1. **Retrieval (Pengambilan Data)**: Data relevan diambil dari basis data internal sebelum mengirim permintaan ke *OpenAI*. Dalam konteks *Skin Analysis*, data

yang diambil adalah katalog produk *Shine2Gether* dari Tabel 3.3 dan Tabel 3.4.

2. **Augmentation (Peningkatan Konteks)**: Data yang diambil ditambahkan ke dalam *system prompt* sebagai *Knowledge Base*, sehingga model *OpenAI* memiliki konteks lengkap tentang produk yang tersedia.

3. **Generation (Generasi Hasil)**: Model *OpenAI* menghasilkan analisis kulit *DAN* rekomendasi produk berdasarkan *knowledge base* tersebut, bukan hanya analisis generik.

Dengan *RAG*, rekomendasi produk selalu relevan dengan *inventory Shine2Gether* yang aktif di *database*, dan dapat diperbarui tanpa perlu mengubah model atau kode *backend*.

A.3.2 Konsolidasi Data untuk Input OpenAI

Tiga jenis data utama dikonsolidasikan sebelum mengirim permintaan ke *OpenAI*:

1. Data Produk (Knowledge Base)

Data produk diambil melalui operasi penggabungan (*joining*) dua tabel relasional. Struktur penyimpanan untuk identitas produk utama dijabarkan pada Tabel 3.3 berikut:

Tabel 3.3. Struktur Tabel *products* (Identitas Produk Utama)

Nama Kolom	**Tipe Data**	**Deskripsi / Fungsi untuk AI**
id	*CHAR(36)*	*Primary Key* (*UUID*)
sku	*VARCHAR(50)*	Kode unik produk (misal: S2G-MSWS-30)
product_name	*VARCHAR(255)*	Nama lengkap produk yang akan direkomendasikan
claims	*JSON*	Klaim utama produk (misal: "Glass Skin Finish")

Informasi teknis mendalam mengenai formulasi produk dijabarkan pada Tabel 3.4 berikut:

Tabel 3.4. Struktur Tabel *prod_shine* (Detail Formulasi)

Nama Kolom	**Tipe Data**	**Deskripsi / Fungsi untuk AI**
id	*CHAR(36)*	*Primary Key* (*UUID*)
product_id	*CHAR(36)*	*Foreign Key* ke Tabel 3.3
key_ingredients	*JSON*	Daftar bahan aktif agar *AI* memahami khasiat produk
problem_targeted	*TEXT*	Target masalah (misal: *Acne*, *Antiaging*) untuk pencocokan dengan hasil analisis wajah
how_to_use	*JSON*	Instruksi pemakaian (Pagi/Malam)
detailed_ingredients	*TEXT*	Komposisi lengkap produk
size_ml	*INT*	Ukuran kemasan produk
key_function	*JSON*	Fungsi utama bahan aktif

2. Logika Seleksi AI Dinamis

Aturan logika *AI* yang aktif diambil dari Tabel 3.7. Aturan ini berisi instruksi spesifik tentang prioritas rekomendasi, aturan *layering* produk, dan panduan penjelasan kandungan aktif.

3. Instruksi Analisis Komprehensif

Instruksi analisis yang lengkap disusun, mencakup:

- Tujuan analisis (validasi kualitas gambar, deteksi kondisi kulit, rekomendasi produk)
- Aturan format keluaran (*JSON* dengan skema terstruktur)
- Penjelasan terminologi *Fitzpatrick skin type* dan bentuk wajah
- Metrik penilaian masalah kulit (0–100 *score* dengan *level* I–V)
- *Heuristik* klasifikasi tipe kulit berdasarkan *sebum* dan hidrasi

A.3.3 Implementasi Prompt Building dengan *llm_handler.py*

Modul *llm_handler.py* mengorkestrasi seluruh proses konsolidasi data. Tiga fungsi utama bekerja bersama.

Fungsi 1: get_product_catalog()

Fungsi ini melakukan *retrieval* data produk dari basis data dengan operasi *JOIN* antara Tabel 3.3 dan Tabel 3.4, seperti ditunjukkan pada Kode 3.3 berikut:

```
def get_product_catalog():
    conn = get_connection()
    with conn.cursor() as cursor:
        cursor.execute("""
            SELECT
                p.sku, p.product_name, p.claims,
                s.size_ml, s.key_function, s.key_ingredients,
                s.problem_targeted, s.how_to_use, s.
    detailed_ingredients
            FROM products p
            LEFT JOIN prod_shine s ON p.id = s.product_id
            WHERE p.brand = 'shine2gether'
        """)
        products = cursor.fetchall()
    conn.close()

    catalog = []
    for p in products:
        catalog.append({
            "sku": p["sku"],
            "name": p["product_name"],
            "claims": json.loads(p["claims"]) if p["claims"] else
    [],
            "size_ml": p.get("size_ml"),
            "key_function": json.loads(p["key_function"]) if p.get
    ("key_function") else [],
            "key_ingredients": json.loads(p["key_ingredients"]) if
     p.get("key_ingredients") else [],
            "problem_targeted": p.get("problem_targeted"),
            "how_to_use": json.loads(p["how_to_use"]) if p.get("
    how_to_use") else [],
            "detailed_ingredients": p.get("detailed_ingredients"),
        })
    return catalog
```

Kode 3.3: Fungsi *Retrieval* Data Produk untuk RAG

Hasil dari fungsi ini adalah struktur *JSON* yang berisi semua produk aktif dengan detail lengkapnya. Contoh output dari katalog produk ditunjukkan pada Kode 3.4 berikut:

```
[
```

```
  {
    "sku": "S2G-MSWS-30",
    "name": "Monster Snow White Serum",
    "claims": ["Brightening", "Anti-spot"],
    "size_ml": 30,
    "key_ingredients": [
      {"name": "Niacinamide", "ppm": "50000"},
      {"name": "Vitamin C", "ppm": "30000"}
    ],
    "problem_targeted": "Brightening, Pigmentation, Pore refining"
   ,
    "how_to_use": [
      {"time": "morning", "instruction": "Apply 2-3 drops"},
      {"time": "evening", "instruction": "Apply 2-3 drops"}
    ]
  },
  ...
]
```

Kode 3.4: Contoh Output Katalog Produk (Partial)

Fungsi 2: get_ai_selection_logic()

Fungsi ini mengambil aturan logika *AI* yang sedang aktif. Aturan disimpan dalam Tabel 3.7 dan dapat diperbarui oleh administrator tanpa perlu mengubah kode, seperti ditunjukkan pada Kode 3.5 berikut:

```
def get_ai_selection_logic():
    conn = get_connection()
    with conn.cursor() as cursor:
        cursor.execute("""
            SELECT rule_text FROM settings_ai_selection_logic
            WHERE is_active=TRUE
            ORDER BY id DESC LIMIT 1
        """)
        row = cursor.fetchone()
    conn.close()
    return row['rule_text'] if row else ""
```

Kode 3.5: Fungsi *Retrieval* Logika AI Dinamis

Fungsi 3: build_face_analysis_prompt()

Ini adalah fungsi inti yang mengkonsolidasikan *semua* data menjadi satu *system prompt* yang kohesif. Fungsi menggabungkan:

- Instruksi analisis komprehensif (TUJUAN, ATURAN KELUARAN,

penjelasan *Fitzpatrick*, skema *JSON*)

- Logika seleksi *AI* dinamis (dari *get_ai_selection_logic()*)
- Katalog produk lengkap (dari *get_product_catalog()*)

Hasil akhirnya adalah satu *system prompt* tunggal yang dikirim ke *OpenAI* bersama dengan gambar pengguna. Fungsi ini ditunjukkan pada Kode 3.6 berikut:

```
def build_face_analysis_prompt(product_catalog):
    selection_logic = get_ai_selection_logic()

    return (
        "TUJUAN: Terima 1 foto wajah (frontal, tanpa filter) --> validasi kualitas --> "
        "deteksi fitur & kondisi kulit --> nilai masalah per-indikator --> keluarkan HASIL "
        "DALAM JSON (sesuai SKEMA) + saran perawatan + rekomendasi produk SHiNE2GeTHER "
        "dari PRODUCT_CATALOG di bawah.\n\n"

        "Jika tersedia, gunakan data history analisis kulit (history JSON) untuk "
        "membandingkan perubahan kondisi kulit. Berikan insight perbandingan pada bagian "
        "\"notes\" di hasil JSON.\n\n"

        "ATURAN KELUARAN:\n"
        "    Keluarkan JSON valid persis mengikuti SKEMA.\n"
        "    Semua deskripsi & saran dalam Bahasa Indonesia.\n"
        "    Untuk setiap indikator, score 0  100  , level  I V , dan confidence 0  1  .\n\n"

        "PENJELASAN FITZPATRICK SKIN TYPE:\n"
        "- Type I: Sangat Pucat, undertone Cool/Pink/Neutral, sering di Eropa Utara.\n"
        "- Type II: Putih Terang, undertone Peach/Pink/Neutral, umum di Eropa Utara & Asia Timur.\n"
        "- Type III: Kuning Langsat, undertone Warm Yellow/Olive, sering di Asia Timur & Asia Tenggara.\n"
        "- Type IV: Sawo Matang, undertone Golden/Olive Neutral, umum di Asia Selatan, Timur Tengah, Amerika Latin.\n"
        "- Type V: Cokelat India, undertone Golden Brown/Red Brown , sering di Asia Selatan, Afrika Utara.\n"
```

```
        "- Type VI: African Deep, undertone Deep Neutral/Blue/Red, umum di Afrika.\n\n"

        "PENJELASAN MASALAH & TIPE BIBIR:\n"
        "- Bentuk bibir: Full Lips, Thin Lips, Top-Heavy Lips, Bottom-Heavy Lips, Heart-Shaped Lips, Round Lips, Wide Lips, Downturned Lips.\n"
        "- Kondisi kulit bibir: Normal, Kering & Dehidrasi, Pecah-pecah, Pigmentasi Gelap, Pucat, Iritasi, Cold Sores, Wrinkles, Cheilitis, Sensitif.\n\n"

        "SKEMA \\textit{JSON} (WAJIB):\\n"
        "{\n"
        '  "quality": {"notes": ""},\n'
        '  "identitas": {...},\n'
        '  "lips": {...},\n'
        '  "indices": {...},\n'
        '  "hydration_moisture_ratio": 0,\n'
        '  "skin_type_summary": {...},\n'
        '  "top_issues": [...],\n'
        '  "care_advice": {...},\n'
        '  "recommendations": [...]\n'
        "}\n\n"

        f"{selection_logic}\n\n"

        "\\textit{PRODUCT_CATALOG}:\\n"
        f"{json.dumps(product_catalog, ensure_ascii=False, indent=2)}\n\n"

        "Berikan analisis berdasarkan foto yang dikirim. Jangan tambahkan penjelasan lain "
        "di luar format JSON di atas.\n\n"
        "DISCLAIMER: Hasil analisis AI ini hanya sebagai referensi edukasi, bukan diagnosis medis."
    )
```

Kode 3.6: Fungsi *Build Prompt* Dinamis dengan RAG

A.3.4 Alur Proses Lengkap: Dari Gambar hingga Hasil

1. Pengambilan Gambar (Frontend)

Pengguna mengambil foto wajah yang telah divalidasi oleh *face-api.js*

(pencahayaan, *angle*, jumlah wajah). Gambar dikirim ke *backend* sebagai pengkodean *base64*.

2. Orkestrasi Backend (Request Handling)

Backend menerima permintaan analisis dan melakukan tahap *retrieval*:

1. Validasi format *file* (pemeriksaan *MIME type*)
2. Panggil *llm_handler.get_product_catalog()* untuk mengambil semua produk S2G dari basis data
3. Panggil *llm_handler.build_face_analysis_prompt(product_catalog)* untuk menyusun *system prompt* lengkap dengan semua instruksi dan *knowledge base*

3. Pengiriman ke OpenAI dengan RAG

Permintaan dikirimkan ke *OpenAI* dengan struktur yang ditunjukkan pada Kode 3.7 berikut:

```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    \textit{messages}=[
        {
            "role": "system",
            "content": build_face_analysis_prompt(product_catalog)
            # ^ Ini adalah SYSTEM PROMPT lengkap dengan:
            #   - Instruksi analisis detail
            #   - Skema JSON output
            #   - Logika AI dinamis
            #   - KNOWLEDGE BASE (katalog produk dari database)
        },
        {
            "role": "user",
            "content": [
                {"type": "text", "text": "Analisis kondisi kulit ini."},
                {"type": "image_url", "image_url": {"url": base64_image}}
                # ^ Gambar yang dikirim pengguna
            ]
        }
    ],
    \textit{response_format}={"type": "json_object"}
    # ^ Memastikan output dalam format JSON sesuai SKEMA
```

```
)
```

Kode 3.7: Request OpenAI dengan RAG

4. Response OpenAI

OpenAI mengembalikan respons *JSON* yang mencakup:

- Analisis identitas (*umur, jenis kelamin, Fitzpatrick type, bentuk wajah*)
- Analisis detail kulit (*wrinkles, sebum, pores, acne, spots*, dll)
- Penilaian masalah utama (*top issues*) dengan *score* dan *confidence*
- Saran perawatan (*care advice*)
- Rekomendasi produk dari PRODUCT_CATALOG yang relevan

5. Penyimpanan dan Agregasi

Backend menyimpan hasil *JSON* ke basis data menggunakan skema *Header-Detail*:

- Tabel 3.5: Menyimpan rangkuman agregat dari 5 analisis terakhir (*summary_all*) menggunakan logika *FIFO Eviction*
- Tabel 3.6: Menyimpan hasil analisis *raw* (*ai_result*) dan ringkasan sesi (*summarize_this*)

6. Visualisasi Frontend

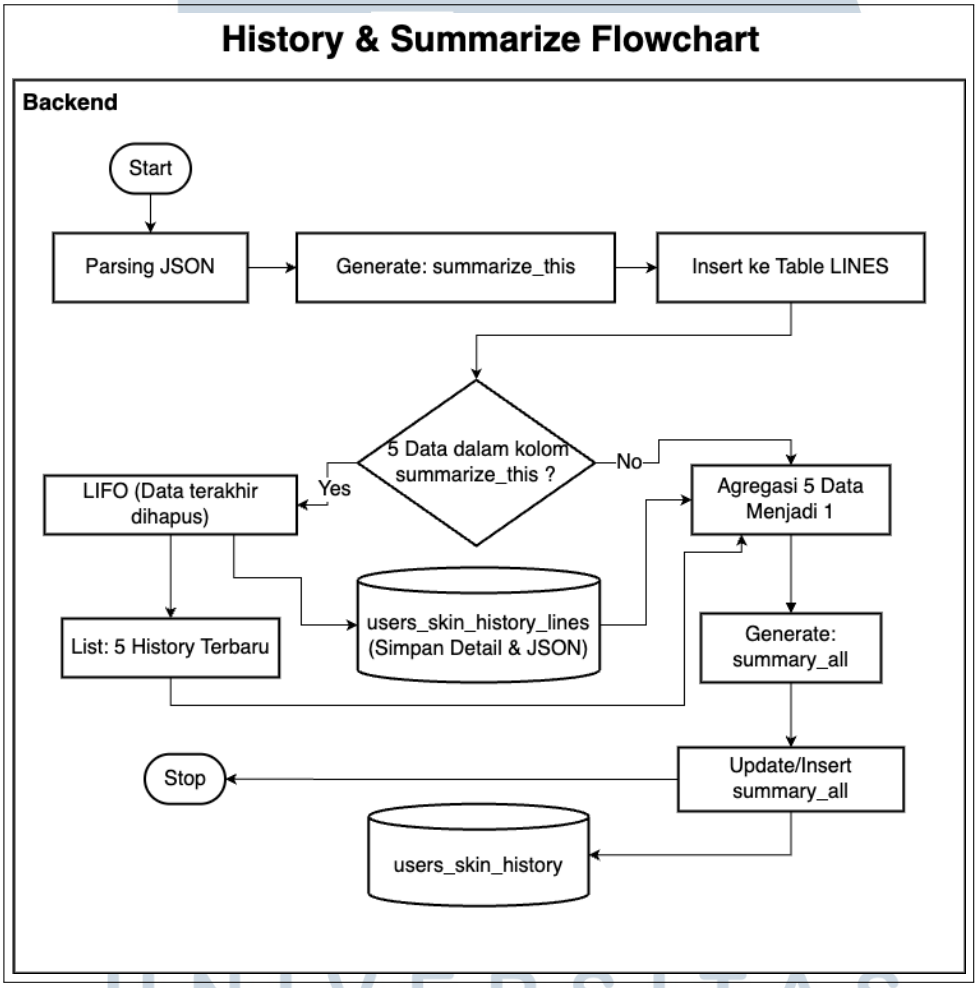
Data hasil analisis *JSON* dikembalikan ke *frontend* dan divisualisasikan menjadi grafik batang interaktif, saran perawatan, dan rekomendasi produk.

A.4 Keunggulan Arsitektur RAG dalam Sistem Ini

1. **Dinamis & Fleksibel**: Produk dapat ditambah/diubah di basis data, dan rekomendasi *AI* otomatis akan berubah tanpa perlu *deployment* ulang.
2. **Akurat & Relevan**: *AI* tidak merekomendasikan produk yang tidak ada di *inventory*, selalu relevan dengan *SKU* yang tersedia.
3. **Dapat Diinterpretasi**: Setiap rekomendasi produk didukung oleh penjelasan detail (*ingredients*, *problem_targeted*, *how_to_use*).
4. **Dapat Dikustomisasi**: Administrator dapat mengubah logika seleksi melalui antarmuka *Sidebar_Settings* tanpa mengubah model *OpenAI*.

A.5 Proses Penyimpanan dan Visualisasi Hasil

Hasil respon dari *OpenAI* berupa *JSON String* kemudian di*parsing* dan disimpan ke dalam basis data. Penyimpanan menggunakan skema *Header-Detail*, dimana komponen *header* berfungsi sebagai penyimpan rangkuman *agregat*, sedangkan komponen *detail* menyimpan riwayat per sesi. Alur logika penyimpanan dan *agregasi* data digambarkan pada Gambar 3.4 berikut.



Gambar 3.4. Alur logika penyimpanan data *Header-Detail* dan agregasi *Rolling Summary*

Struktur penyimpanan ini menerapkan logika *Rolling Summary* dengan mekanisme sebagai berikut:

1. Setiap kali analisis baru dilakukan, hasilnya disimpan ke 3.6 pada kolom *summarize_this*.
2. Sistem kemudian mengambil 5 data analisis terakhir (*Latest-5*) dari 3.6.

3. Kelima data tersebut dirangkum ulang menjadi satu kesimpulan tren kondisi kulit, lalu disimpan ke 3.5 pada kolom *summary_all*.

4. Jika terdapat data ke-6, data terlama tidak akan diikutsertakan dalam pembentukan rangkuman utama (*FIFO Eviction for Summary*), memastikan informasi profil pengguna selalu relevan dengan kondisi terkini.

Dua tabel utama yang digunakan dalam skema penyimpanan ini adalah:

1. Tabel 3.5: Berfungsi sebagai komponen *header* profil riwayat yang menyimpan rangkuman agregat. Struktur tabel ditunjukkan pada Tabel 3.5 berikut.

2. Tabel 3.6: Berfungsi menyimpan setiap sesi analisis yang dilakukan pengguna. Struktur tabel ditunjukkan pada Tabel 3.6 berikut.

Tabel 3.5. Struktur Tabel *users_skin_history* (*Header* / Agregat)

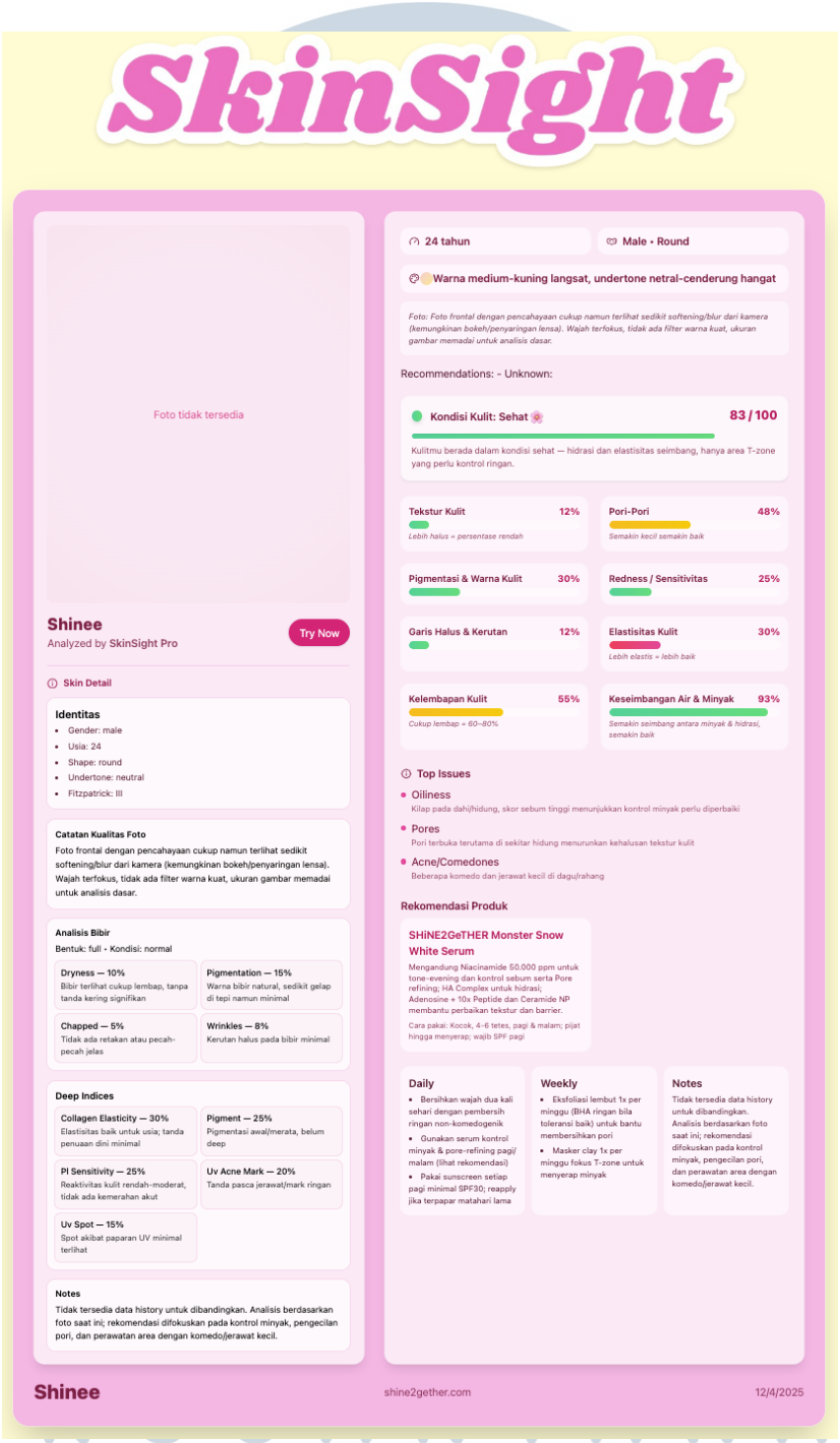
Nama Kolom	**Tipe Data**	**Deskripsi**
id	VARCHAR(64)	*Primary Key* (*UUID*)
user_id	VARCHAR(64)	*Foreign Key* ke tabel *users*
summary_all	TEXT	Kesimpulan tren kulit yang diolah dari 5 riwayat terakhir.
created_at	DATETIME	Waktu pembaharuan terakhir

Tabel 3.6. Struktur Tabel *users_skin_history_lines* (*Detail* / Sesi)

Nama Kolom	**Tipe Data**	**Deskripsi**
id	VARCHAR(36)	*Primary Key* (*UUID*) detail *history*
parent_id	VARCHAR(64)	*Foreign Key* ke komponen *header*
ai_result	LONGTEXT	Data mentah (*Raw JSON*) hasil analisis lengkap dari *OpenAI*
summarize_this	TEXT	Rangkuman Sesi Hasil analisis spesifik pada saat pengambilan gambar ini saja.

Setelah data tersimpan dan rangkuman diperbarui, informasi dikirim kembali ke sisi pengguna (*frontend*). Data *JSON* tersebut kemudian divisualisasikan menjadi grafik

batang (*metric bars*) yang interaktif. Tampilan hasil akhir analisis dapat dilihat pada Gambar 3.5 berikut.



Gambar 3.5. Visualisasi hasil analisis kulit dan rekomendasi produk pada sisi *Frontend*

B Implementasi Sisi Administrator (Admin Side)

Halaman administrator berfungsi sebagai pusat kendali data (*Control Center*). Fitur-fitur di halaman ini memungkinkan pengelolaan *Knowledge Base* yang digunakan oleh *AI*, serta pengaturan logika seleksi produk tanpa harus mengubah kode asli program.
Implementasi halaman administrator dibagi menjadi dua modul utama: manajemen katalog produk dan konfigurasi logika AI.

B.1 Manajemen Katalog Produk (Knowledge Base Management)

Modul ini memasukkan dan memperbarui data produk. Sebagai sumber data utama bagi arsitektur *RAG*, kelengkapan informasi yang dikelola pada modul ini sangat menentukan kualitas dan relevansi saran produk yang dihasilkan oleh *AI*.
Antarmuka *frontend* dikembangkan menggunakan *React* (*file Sidebar_EditProducts.tsx*). Formulir input dirancang secara dinamis menyesuaikan merek produk. Khusus untuk produk *Shine2Gether*, formulir menampilkan kolom detail teknis seperti *Key Ingredients*, *Problem Targeted*, dan *Claims*. Implementasi formulir dinamis ditunjukkan pada Kode 3.8 berikut.

```
{selectedBrand === 'shine2gether' && (
  <>
    <div className="mb-2">
       <label>Key Ingredients (JSON: name, ppm)</label>
       {/* Input dinamis untuk bahan aktif */}
       {form.key_ingredients.map((ing: any, i: number) => (
          <div key={i} className="flex gap-2 mb-1">
             <input
               placeholder="Name (e.g. Niacinamide)"
               value={ing.name}
               onChange={...}
             />
             <input
               placeholder="PPM (e.g. 50000)"
               value={ing.ppm}
               onChange={...}
             />
          </div>
       ))}
    </div>

```

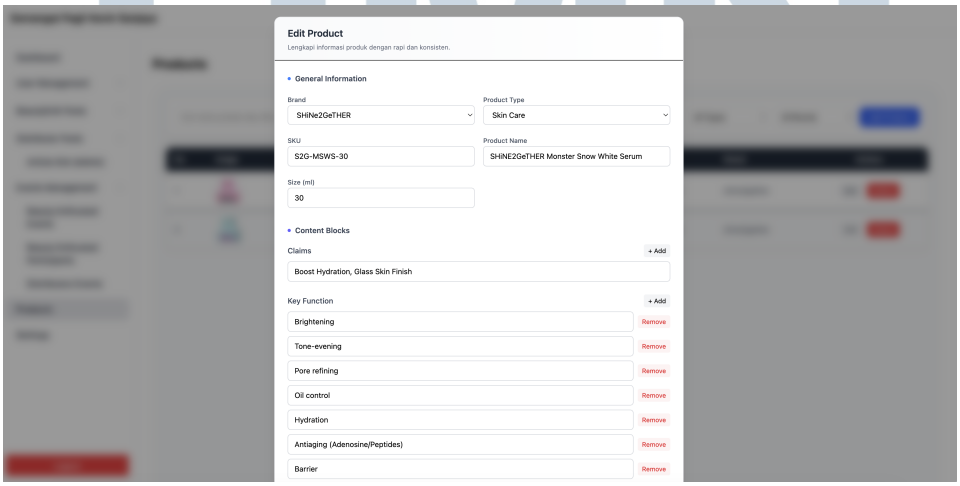
```
22      <div className="mb-2">
23          <label>Problem Targeted (Comma separated)</label>
24          <textarea
25            className="border w-full px-2 py-1 rounded"
26            placeholder="e.g. Acne, Brightening, Antiaging"
27            value={form.problem_targeted || ''}
28            onChange={e => handleFormChange('problem_targeted', e.
    target.value)}
29          />
30       </div>
31    </>
32  )}
```

Kode 3.8: Formulir Dinamis Input Produk pada *Sidebar_EditProducts.tsx*

Di sisi *backend* (*file admin_products.py*), penyimpanan data ditangani ke dalam basis data relasional. Data yang diinput oleh administrator dipetakan ke dalam dua tabel yang telah dijelaskan sebelumnya:

- Data identitas dasar disimpan sesuai struktur pada Tabel 3.3.
- Data formulasi mendalam disimpan sesuai struktur pada Tabel 3.4.

Mekanisme ini memastikan integritas data, di mana saat administrator menghapus produk, otomatis dilakukan *Cascade Delete* untuk membersihkan data di kedua tabel tersebut. Tampilan antarmuka manajemen produk dapat dilihat pada Gambar 3.6 berikut.



Gambar 3.6. Antarmuka input data produk dengan detail parameter untuk AI

B.2 Konfigurasi Logika Seleksi AI (Dynamic AI Rules)

Salah satu keunggulan sistem adalah fleksibilitas logika *AI*. Administrator dapat mengubah aturan seleksi (*selection logic*) atau menyempurnakan instruksi (*prompt*) melalui halaman *settings* tanpa perlu melakukan *deployment* ulang.
Seluruh aturan logika disimpan dalam tabel khusus bernama *settings_ai_selection_logic*. Struktur penyimpanan ini dirancang untuk menampung teks instruksi panjang (*prompting*) yang akan disuntikkan ke model *OpenAI*. Struktur tabel ditunjukkan pada Tabel 3.7 berikut.

Tabel 3.7. Struktur Tabel *settings_ai_selection_logic*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	BIGINT UNSIGNED	*Primary Key* sebagai identitas unik aturan dengan *AUTO_INCREMENT*.
rule_name	VARCHAR(100)	Nama aturan (misal: "Default").
rule_text	TEXT	Instruksi lengkap (*System Prompt*) yang berisi logika pemilihan produk, aturan *layering*, dan penjelasan kandungan aktif.
is_active	TINYINT(1)	Penanda status aktif (1 = Aktif, 0 = Non-aktif). *Default* 1. Hanya satu aturan yang boleh aktif dalam satu waktu.

Pada *file Sidebar_Settings.tsx*, administrator mengelola data pada Tabel 3.7. Data aturan yang bertanda *is_active* = 1 nantinya diambil oleh *backend* dan digabungkan ke dalam *System Prompt OpenAI*. Implementasi *endpoint backend* untuk mengambil konfigurasi ditunjukkan pada Kode 3.9 berikut.

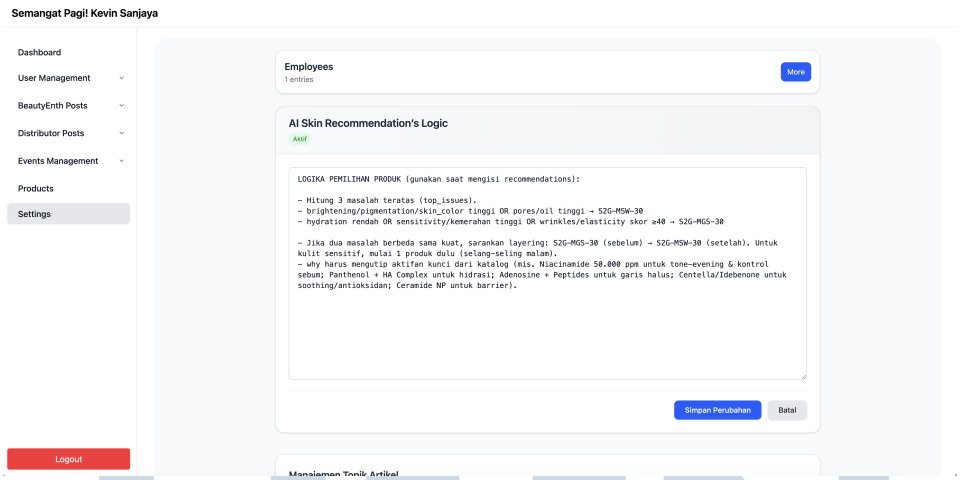
```
# Get all AI selection logic rules
@admin_settings.route('/admin/ai-selection-logic', methods=['GET'])
@require_admin
def get_ai_selection_logic_list():
    """ Mengambil semua aturan logika AI dari database """
    conn = get_connection()
    try:
        with conn.cursor() as cursor:
            # Data diambil untuk ditampilkan di tabel Admin
            cursor.execute(""" SELECT * FROM settings_ai_selection_logic ORDER BY id DESC """)
```

```
            rows = cursor.fetchall()

        return jsonify({'rules': rows}), 200
    except Exception as e:
        return jsonify({'rules': [], 'error': 'database error'}),
500
```

Kode 3.9: *Endpoint* Manajemen Logika AI pada *admin_settings.py*

Dengan fitur ini, apabila hasil analisis *AI* dirasa kurang akurat atau terdapat perubahan strategi rekomendasi produk, administrator dapat mengubah isi kolom *rule_text* melalui halaman ini. Tampilan halaman konfigurasi logika *AI* dapat dilihat pada Gambar 3.7 berikut.



Gambar 3.7. Antarmuka konfigurasi aturan logika AI (*AI Selection Logic*)

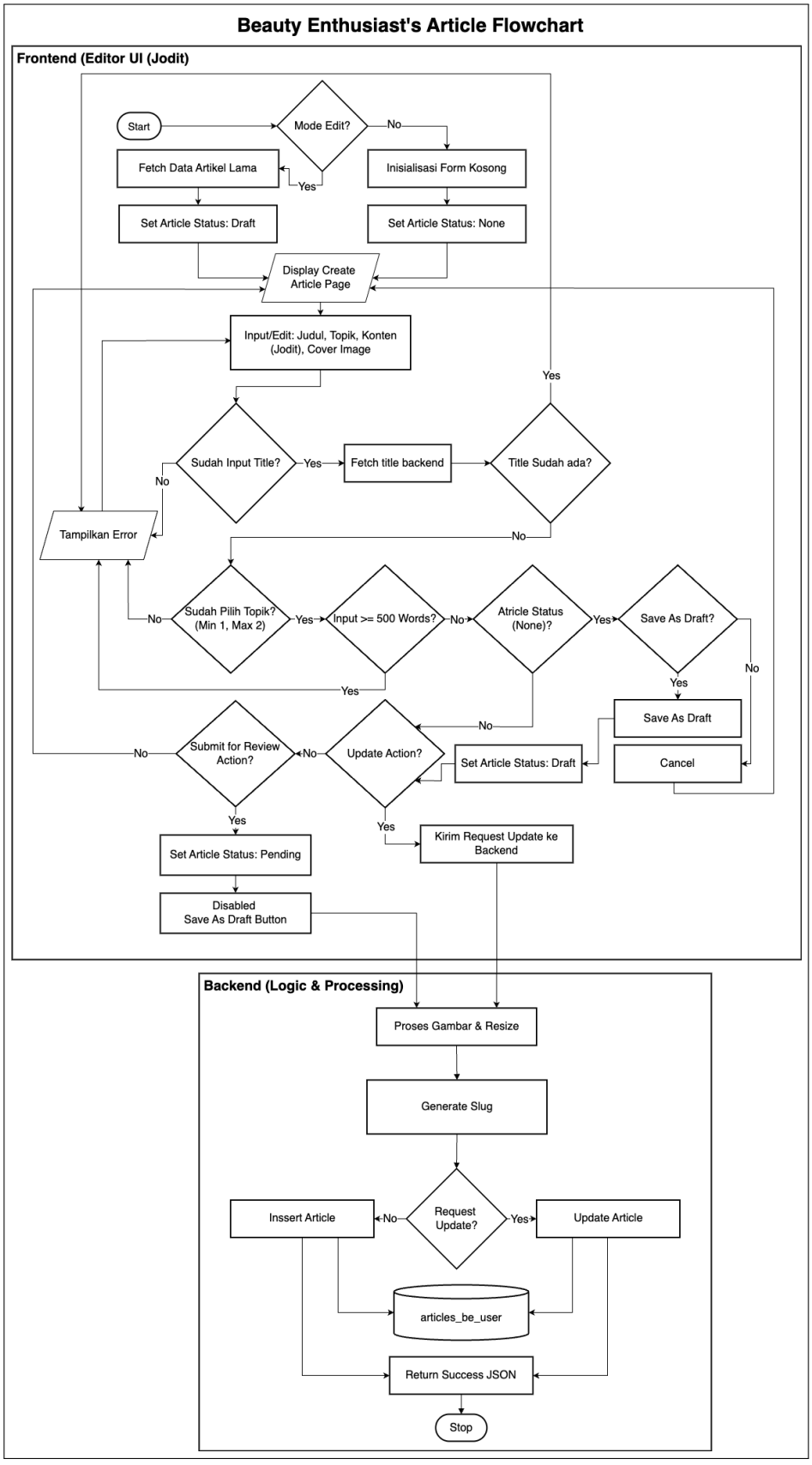
3.3.2 Implementasi Fitur Artikel Komunitas (Beauty Enthusiast Articles)

Fitur Artikel Komunitas merupakan wadah bagi pengguna dengan peran *Beauty Enthusiast* untuk berbagi wawasan, *tips*, dan ulasan produk. Berbeda dengan fitur *blog* biasa, fitur ini menerapkan alur moderasi konten (*Content Moderation*) untuk menjaga kualitas informasi yang dipublikasikan.

Implementasi fitur ini dibagi menjadi dua perspektif utama, yaitu sisi penulis (*User Side*) yang menangani pembuatan konten, dan sisi administrator (*Admin Side*) yang menangani validasi konten.

A Implementasi Sisi Pengguna (Article Creation & Validation)

Pada sisi pengguna, antarmuka utama adalah halaman *Article Editor* yang mengintegrasikan *library Jodit Editor*. Sistem ini dirancang untuk memfasilitasi pengguna dalam membuat konten kaya *(rich text)* tanpa memerlukan pengetahuan teknis *HTML*. Alur logika validasi dan penyimpanan data digambarkan pada Gambar 3.8.

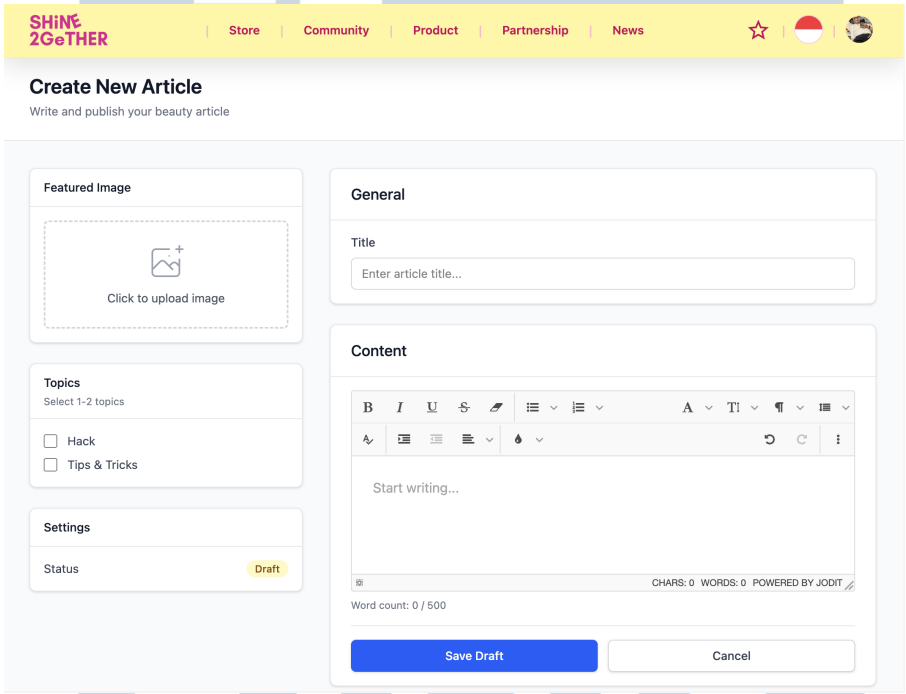


Gambar 3.8. Alur Logika Pembuatan dan Validasi Artikel pada Sisi Pengguna

Implementasi alur logika Pada Gambar 3.8 dibagi menjadi tahapan inisialisasi, penyimpanan, dan hasil akhir.

A.1 Tampilan Inisialisasi (Create New Article)

Ketika pengguna pertama kali mengakses menu ”Buat Artikel”, sistem akan memuat halaman editor dalam keadaan kosong (*Clean State*). Pada tahap ini, status artikel belum terdefinisi. Pengguna diwajibkan mengisi judul, mengunggah gambar sampul (*featured image*), memilih topik, dan menulis konten utama.
Tampilan awal formulir pembuatan artikel dapat dilihat pada Gambar 3.9.



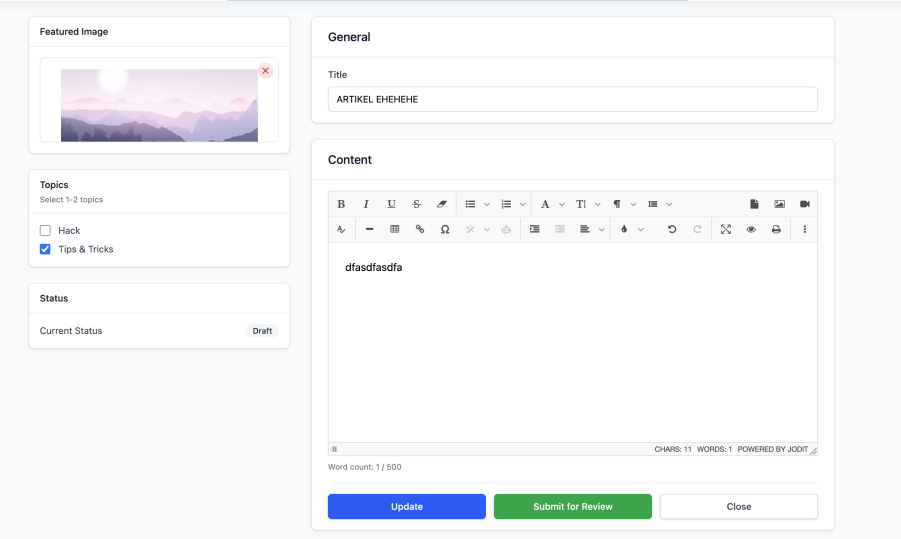
Gambar 3.9. Antarmuka Halaman Pembuatan Artikel Baru (*Form* Kosong)

Pada Gambar 3.9, tombol aksi di bagian bawah secara *default* menampilkan *”Save Draft”*. Sistem validasi akan memblokir proses penyimpanan jika jumlah kata masih di bawah 500 kata atau topik belum dipilih.

A.2 Tampilan Mode Penyuntingan (Saved as Draft)

Setelah pengguna mengisi konten dan menekan tombol *”Save Draft”*, data akan disimpan ke *database* dengan status *’draft’*. Sistem kemudian akan me-*reload* halaman dalam *Mode Edit*.

Perbedaan signifikan pada mode ini terlihat pada status *badge* yang berubah menjadi *"Draft"* dan munculnya opsi tombol baru. Tampilan visualnya ditunjukkan pada Gambar 3.10.



Gambar 3.10. Antarmuka Editor dalam Mode *Edit* dengan Status *Draft*

Seperti terlihat pada Gambar 3.10, terdapat dua tombol aksi utama:

- *Update:* Digunakan untuk menyimpan perubahan terbaru tanpa mengubah status draft.
- *Submit for Review:* Tombol hijau yang berfungsi untuk mengajukan artikel ke admin. Tombol ini mengubah status menjadi *'pending_review'* dan mengunci artikel agar tidak bisa diedit sementara waktu.

A.3 Struktur Penyimpanan Data (Backend)

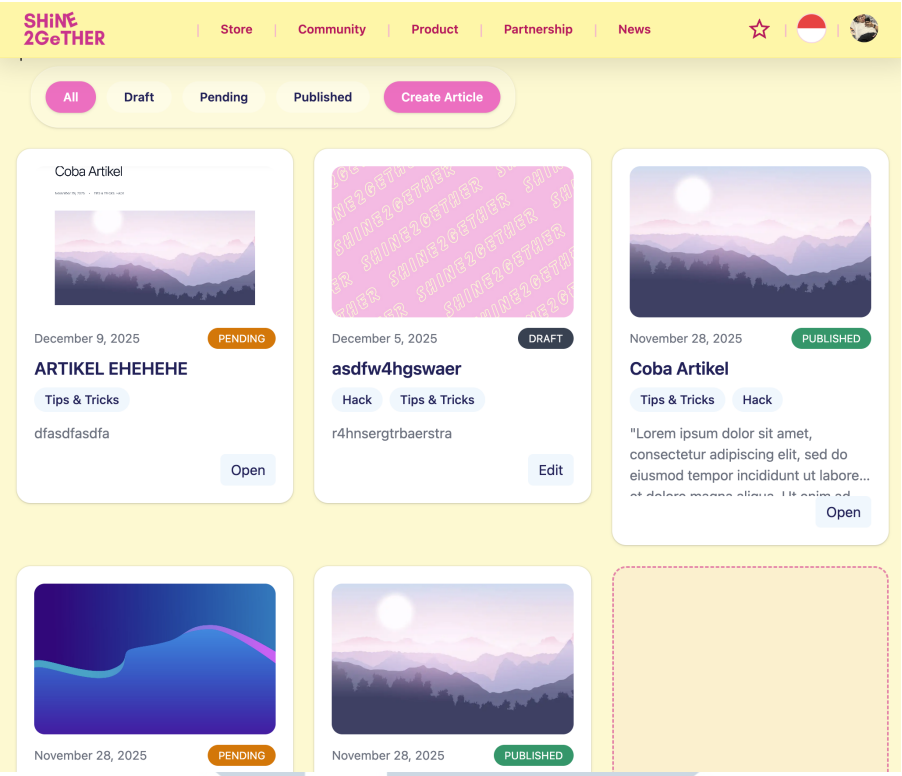
Setiap kali proses *Save Draft*, *Update*, atau *Submit* dilakukan pada langkah di atas, sistem akan menyimpan atau memperbarui data ke dalam tabel *articles_be_user*. Berdasarkan implementasi *database*, tabel ini menggunakan tipe data *JSON* untuk menyimpan *array* topik dan isi konten *(HTML Body)* agar fleksibel. Detail struktur tabel dapat dilihat pada Tabel 3.8.

Tabel 3.8. Struktur Tabel *articles_be_user*

Nama Kolom	**Tipe Data**	**Fungsi**
id	CHAR(36)	*Primary Key (UUID)*
author_id	CHAR(36)	ID Penulis *(Foreign Key)*
title	VARCHAR(255)	Judul artikel
slug	VARCHAR(255)	*URL friendly* dari judul
topics	JSON	Menyimpan *array topics* (e.g., [”Hack”, ”Tips”])
content	JSON	Menyimpan isi artikel (*HTML body* dari *Jodit*)
thumbnail	BLOB	Data *biner* gambar sampul artikel
status	ENUM	*’draft’*, *’pending’*, *’published’*, *’rejected’*

A.4 Tampilan Daftar Artikel (Article Cards)

Setelah data berhasil disimpan dalam struktur tabel di atas, artikel akan ditampilkan dalam bentuk *grid card*. Komponen ini menampilkan ringkasan visual agar menarik minat pembaca. Implementasi tampilan kartu artikel dapat dilihat pada Gambar 3.11.



Gambar 3.11. Tampilan Daftar Artikel dalam Format *Grid Card*

Setiap kartu pada Gambar 3.11 memuat informasi krusial seperti judul artikel, tanggal pembuatan, dan *status badge*. Selain itu, terdapat logika kondisional pada tombol aksi utama:

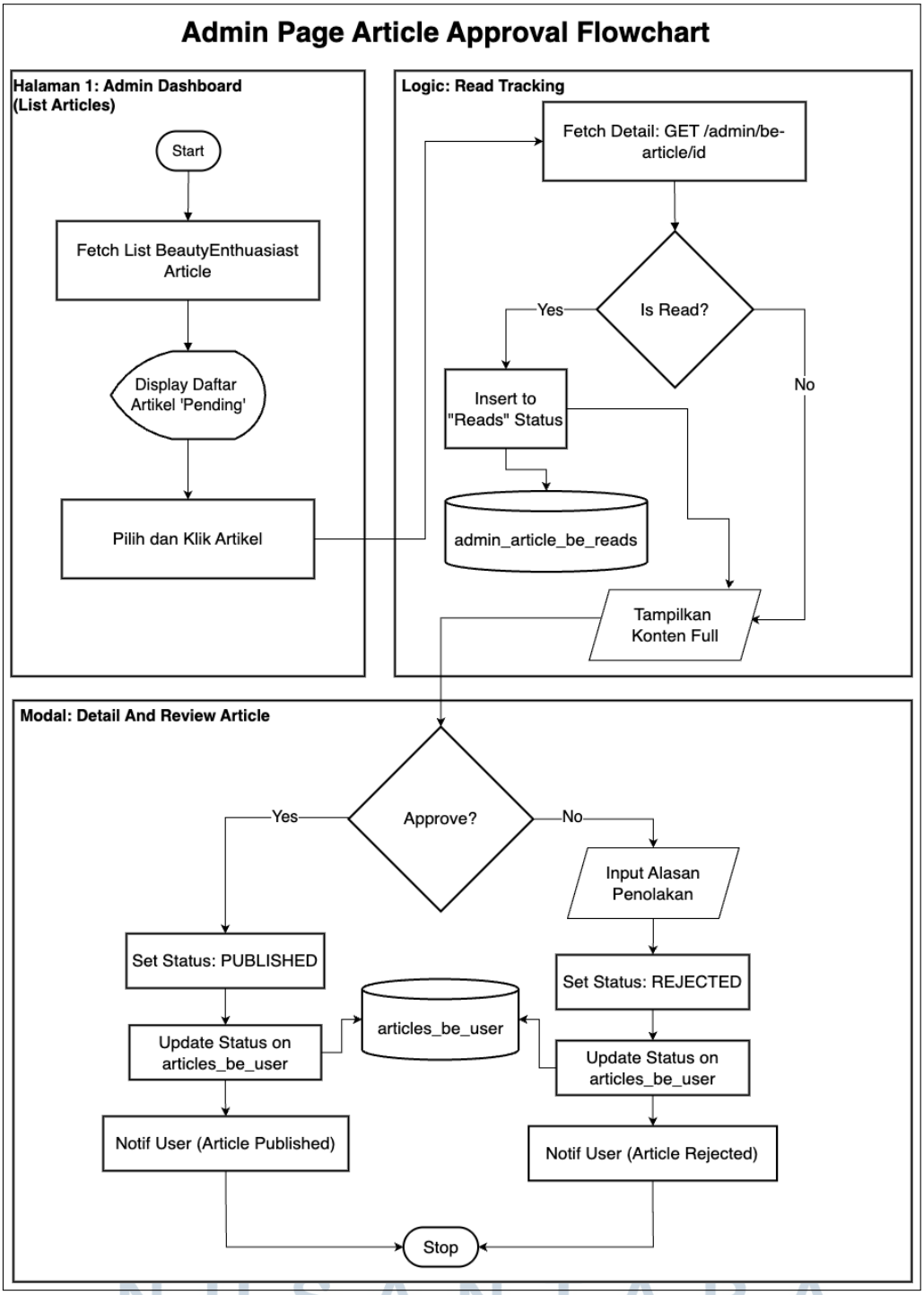
– Status *Published*: Tombol akan berlabel *"Open"*. Tombol ini menggunakan mekanisme *slug-based routing* untuk mengarahkan pengguna ke halaman baca artikel (*View Mode*).

– Status *Draft, Pending, & Rejected*: Tombol akan berlabel *"Edit"*. Tombol ini akan mengarahkan pengguna kembali ke halaman *Article Editor* untuk melanjutkan penulisan atau revisi.

B Implementasi Sisi Administrator (Content Moderation)

Sisi administrator berfungsi sebagai *Gatekeeper* untuk memastikan konten yang tayang aman dan relevan. Berbeda dengan sisi pengguna, administrator tidak membuat konten, melainkan melakukan validasi terhadap artikel yang masuk.
Fitur unggulan pada sisi ini adalah mekanisme *Read Tracking*, di mana sistem melacak apakah admin sudah membaca detail artikel atau belum. Alur kerja lengkap

dari perspektif antarmuka dan logika sistem digambarkan pada Gambar 3.12.

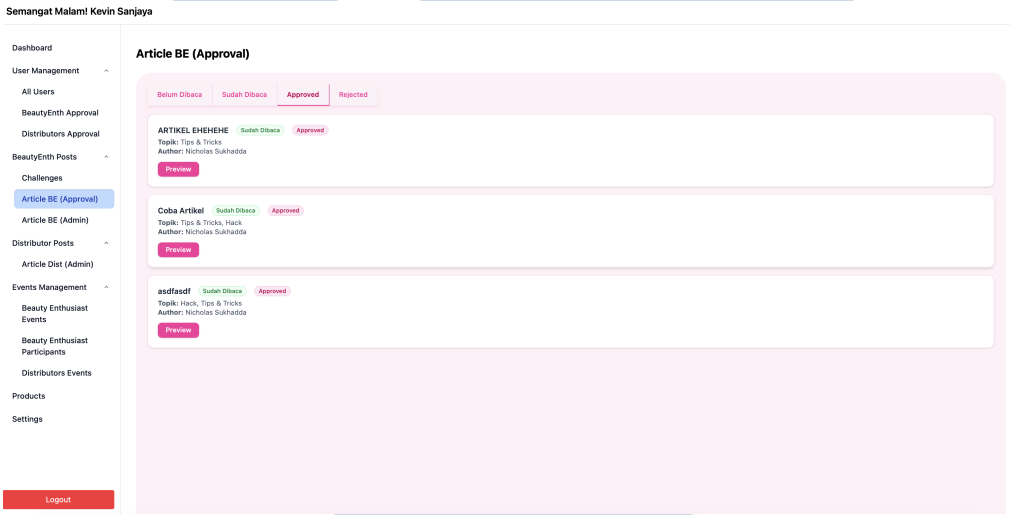


Gambar 3.12. Alur Logika Moderasi Artikel dan Pembagian Halaman pada Sisi Administrator

Berdasarkan Gambar 3.12, implementasi dibagi menjadi dua halaman utama:

B.1 Halaman Dashboard (List View)

Halaman ini merupakan titik masuk (*Entry Point*). Sistem memanggil *API GET /admin/be-articles* untuk mengambil seluruh artikel dengan status *'pending'*. Tampilan antarmuka *dashboard* admin untuk moderasi artikel dapat dilihat pada Gambar 3.13.

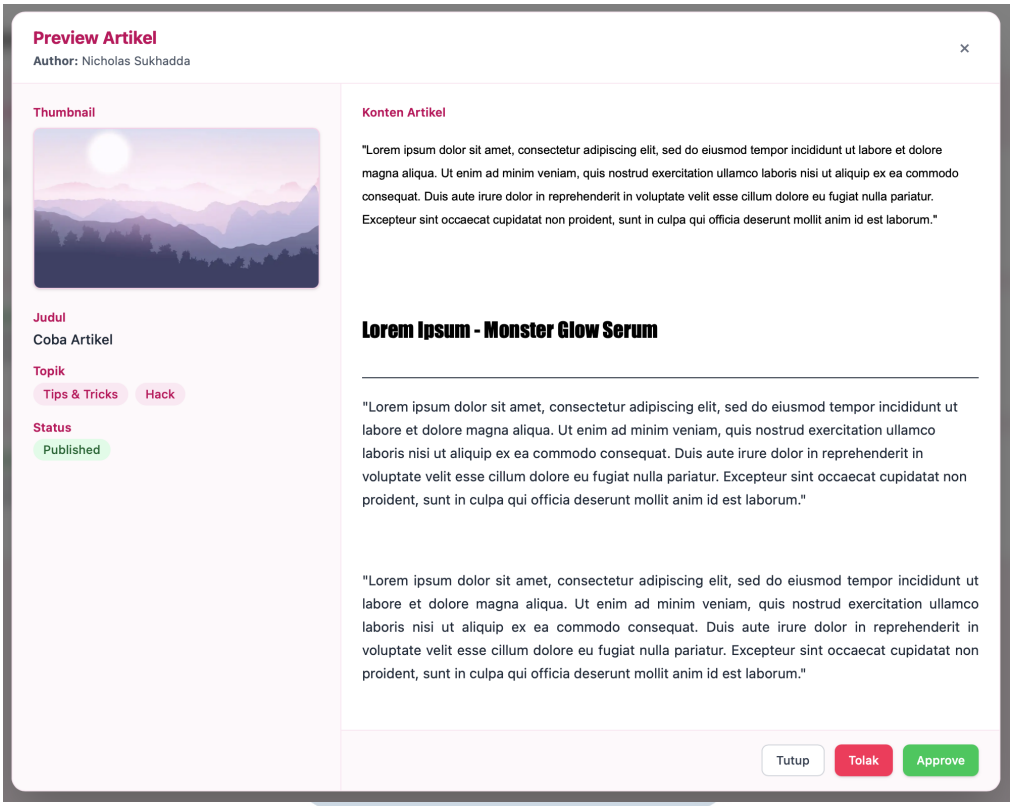


Gambar 3.13. Tampilan Daftar Antrian Artikel pada Dashboard Admin

Seperti terlihat pada Gambar 3.13, admin disajikan tabel yang memuat informasi ringkas: Judul, Penulis, dan Status. Kolom status memudahkan admin memfilter artikel yang membutuhkan tinjauan segera. Admin melakukan aksi pemilihan (*Select Action*) dengan mengklik tombol *"Review"* atau judul artikel untuk masuk ke tahap pemeriksaan rinci.

B.2 Halaman Detail (Review View)

Setelah memilih artikel, admin diarahkan ke halaman detail. Pada tahap ini, konten artikel ditampilkan secara utuh agar admin dapat membaca konteks secara menyeluruh. Implementasi antarmuka halaman tinjauan ditampilkan pada Gambar 3.14.



Gambar 3.14. Antarmuka Halaman Tinjauan dan Validasi Artikel

Pada Gambar 3.14, selain menampilkan konten, sistem menjalankan logika latar belakang (*background logic*) yang krusial:

– *Auto-Read Tracking:* Saat halaman ini dimuat, sistem secara otomatis mencatat bahwa admin telah ”membaca” artikel tersebut di tabel *log*.

– *Moderation Panel:* Di bagian bawah atau samping konten, terdapat panel aksi yang berisi tombol keputusan *(Approve/Reject)* dan kolom *input* catatan *(Notes).*

B.3 Eksekusi Keputusan (Backend Processing)

Keputusan akhir dieksekusi melalui tombol aksi yang akan memanggil *endpoint POST /review*:

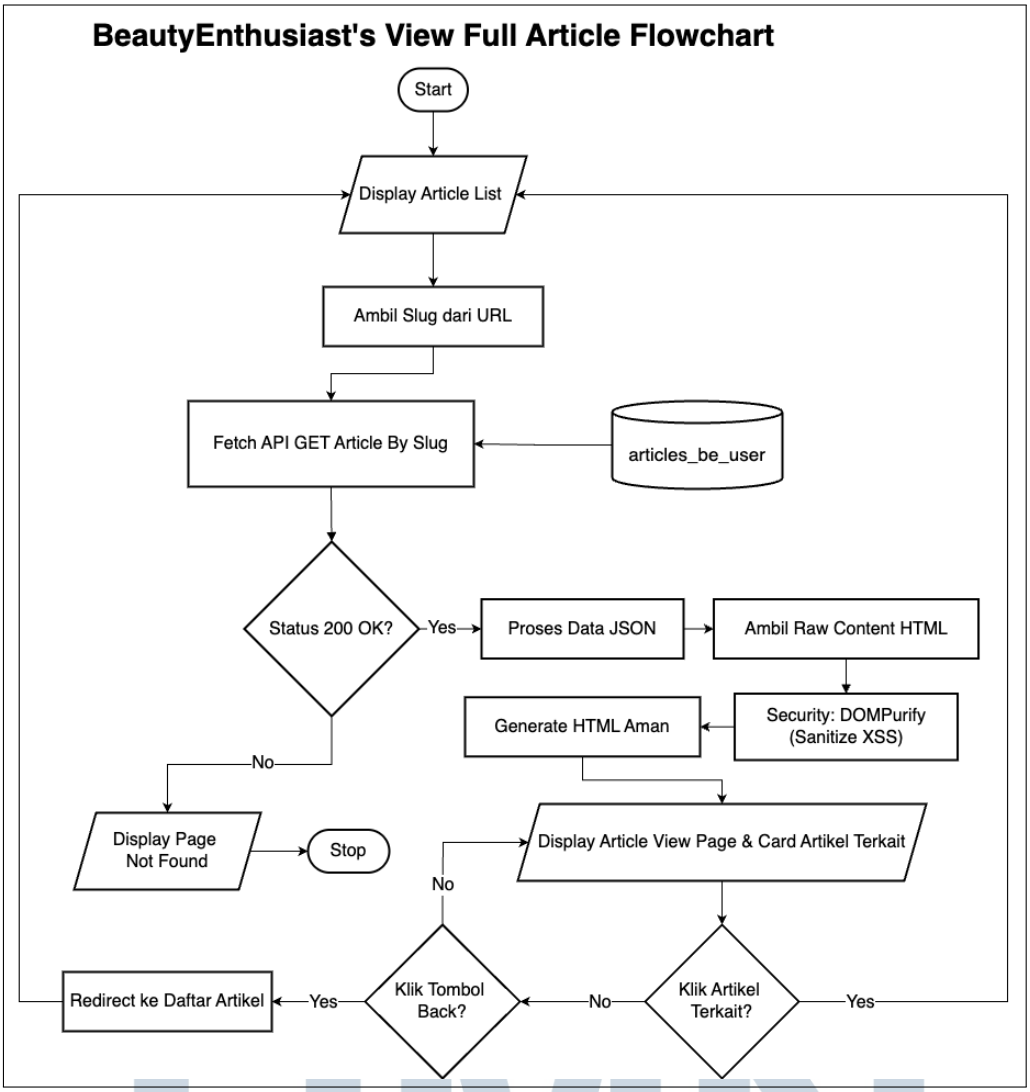
1. *Approve* (Tombol Hijau): Mengubah status menjadi *’published’*. Artikel *published* di sisi publik.

2. *Reject* (Tombol Merah): Mengubah status menjadi *'rejected'*. Saat memilih ini, admin wajib mengisi *form* alasan penolakan pada panel moderasi sebagai umpan balik konstruktif untuk penulis.

Sistem kemudian mengakhiri proses dengan mengirimkan notifikasi otomatis ke penulis artikel terkait hasil moderasi tersebut.

C Implementasi Tampilan Publik (Article Read Mode)

Tahap akhir dari alur publikasi konten adalah penyajian artikel kepada pembaca umum. Artikel yang telah berstatus *'published'* dapat diakses melalui halaman publik yang menerapkan mekanisme *Slug-Based Routing*.
Alur logika teknis mulai dari pengambilan data dari basis data hingga interaksi pengguna digambarkan secara terstruktur pada Gambar 3.15.

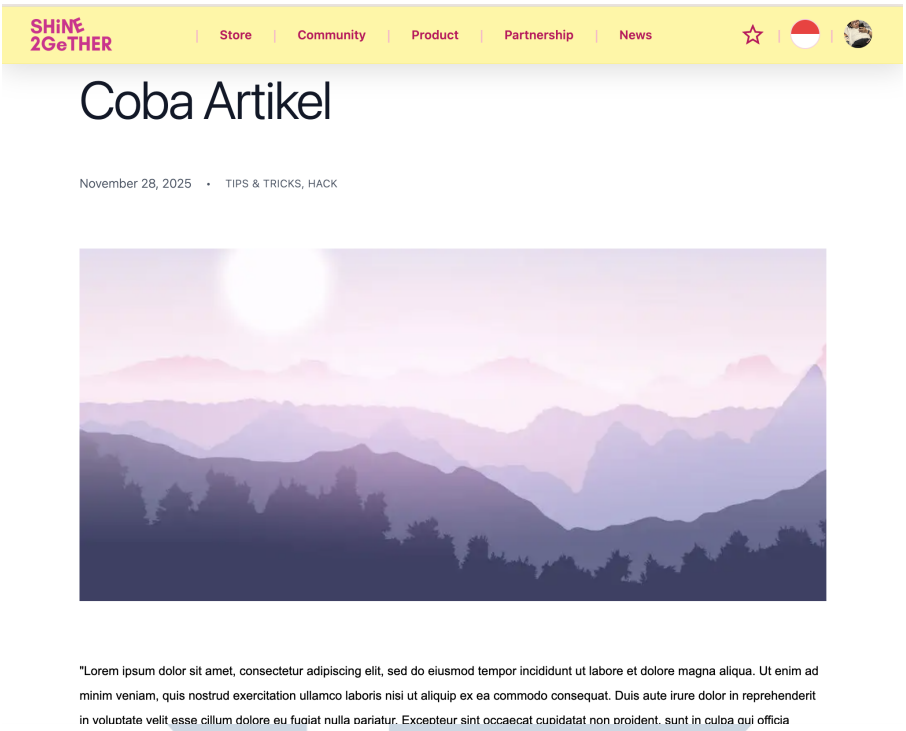


Gambar 3.15. Alur Logika *Rendering* Halaman Artikel Publik

Berdasarkan Gambar 3.15, proses dimulai dengan sistem melakukan *query* ke basis data *MySQL* untuk mencari artikel berdasarkan *slug*. Implementasi visual halaman ini dibagi menjadi tiga segmen utama: *Header*, Konten, dan Navigasi Lanjutan.

C.1 Header dan Metadata Artikel

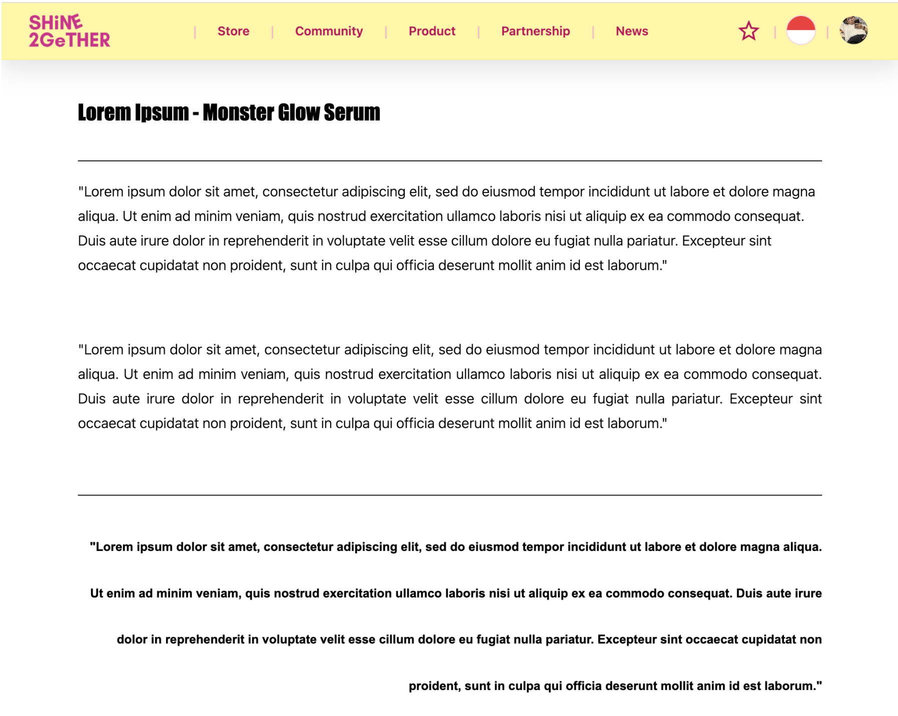
Segmen atas menampilkan identitas artikel secara lengkap, meliputi judul, kategori topik, tanggal publikasi, dan atribusi penulis. Gambar sampul (*Cover Image*) ditampilkan secara penuh untuk menarik perhatian pembaca. Tampilan segmen *header* dapat dilihat pada Gambar 3.16.



Gambar 3.16. Tampilan Segmen *Header* (Judul dan *Metadata*)

C.2 Konten Artikel dan Sanitasi Keamanan

Segmen tengah memuat isi utama artikel. Pada bagian ini, sistem tidak sekadar menampilkan teks, melainkan menerapkan mekanisme keamanan *XSS Prevention*. Implementasi visual konten artikel ditampilkan pada Gambar 3.17.



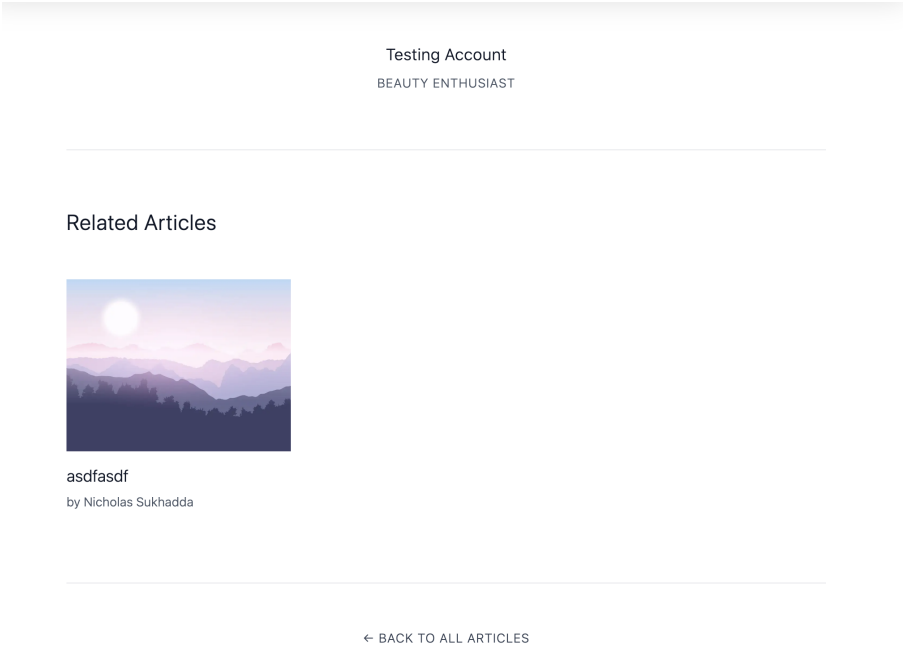
Gambar 3.17. Tampilan Konten Artikel

Sebagaimana terlihat pada Gambar 3.17, format teks (paragraf, tebal, miring) tetap terjaga. Hal ini dimungkinkan berkat penggunaan *library DOMPurify* yang berfungsi:

- *Filtering:* Membersihkan *tag HTML* berbahaya (seperti *script* atau *iframe*) dari data mentah *database*.
- *Rendering:* Menampilkan *HTML* yang sudah "dibersihkan" sehingga aman dari serangan kode jahat namun tetap nyaman dibaca.

C.3 Artikel Terkait (Related Articles)

Di bagian bawah halaman, sistem menyajikan rekomendasi artikel lain untuk meningkatkan durasi kunjungan pengguna. Fitur ini bekerja dengan memfilter artikel lain yang memiliki kesamaan topik dengan artikel yang sedang dibaca. Tampilan segmen rekomendasi artikel dapat dilihat pada Gambar 3.18.



Gambar 3.18. Komponen Artikel Terkait *(Related Articles)*

3.3.3 Implementasi Fitur *Season* dan *Challenge* Beauty Enthusiast

Implementasi fitur ini merupakan komponen fundamental dari strategi *Gamification* pada website *Shine2Gether*. Modul ini dirancang sebagai ekosistem tertutup yang melibatkan interaksi dua arah antara Administrator (sebagai pengelola kompetisi) dan Pengguna (sebagai peserta).

A Prinsip Gamifikasi dan Target Implementasi

Sebelum membahas implementasi teknis, perlu diketahui bahwa fitur *Season* dan *Challenge* didesain berdasarkan tiga prinsip inti *gamification*:

1. ***Progress Visibility*** **(Visibilitas Kemajuan)**—Pengguna harus selalu tahu kemana mereka bergerak. Implementasi sistem menyediakan *dashboard real-time* yang menampilkan akumulasi poin, peringkat (*ranking*), dan status setiap tantangan yang telah/sedang dikerjakan melalui Tabel 3.11 dan 3.13.

2. ***Challenge Variability*** **(Variasi Tantangan)**—Setiap tantangan dirancang berbeda agar tidak membosankan. Sistem memungkinkan *admin* membuat skema input dinamis (teks, *upload* foto, esai) tanpa perlu mengubah kode *backend*. Skema disimpan sebagai *JSON* dalam kolom *participant_form_schema* pada Tabel 3.10.

3. ***Community Competition*** **(Kompetisi Komunitas)**—Papan peringkat

memotivasi pengguna untuk terus berpartisipasi. Data pencapaian pengguna terkumpul di Tabel 3.13 dan dapat divisualisasikan menjadi *leaderboard*.

A.1 Target Capaian Fitur

- **Engagement:** Minimal 40% pengguna aktif mengikuti satu *challenge* per musim dalam kuartal pertama implementasi.
- **Retention:** Pertahankan 70% pengguna yang sudah mencoba agar tetap aktif di bulan berikutnya.
- **Satisfaction:** Tingkat kepuasan moderasi (*fairness* dalam penerimaan/penolakan) minimal 4/5 dalam survei pengguna.
- **Conversion:** Minimal 25% pengguna yang menyelesaikan tantangan melakukan pembelian produk yang direferensikan dalam *challenge* tersebut.

A.2 Cara Mengukur Capaian

Sistem *database* dirancang untuk mencatat semua aktivitas sehingga dapat dianalisis:

1. **Participation Metrics:** *Database* mencatat setiap pengiriman di Tabel 3.11. *Query* sederhana seperti dalam Kode 3.10 memberikan jumlah peserta unik per tantangan.
2. **Approval vs. Rejection Ratio:** Kolom *status* pada Tabel 3.11 mencatat keputusan *admin* (*'pending'*, *'approved'*, *'rejected'*). Analisis distribusi status mengungkap apakah tantangan terlalu sulit/mudah berdasarkan rasio persetujuan.
3. **Point Accumulation Trends:** Tabel 3.13 merekam setiap pemberian poin dengan *timestamp completed_at*. Visualisasi *time-series* memperlihatkan tren keterlibatan sepanjang musim.
4. **Fairness Feedback:** Admin dapat mencatat alasan penolakan di sistem, sehingga pengguna yang ditolak dapat memperbaiki kiriman mereka. Data ini dianalisis untuk perbaikan musim berikutnya.

```sql
SELECT
    c.id, c.title,
    COUNT(DISTINCT p.user_id) AS total_participants,
    SUM(CASE WHEN p.status='approved' THEN 1 ELSE 0 END) AS
    approved_count,
    SUM(CASE WHEN p.status='rejected' THEN 1 ELSE 0 END) AS
    rejected_count
FROM beautyenth_challenges c
LEFT JOIN beautyenth_challenge_participants p ON c.id = p.
    challenge_id
GROUP BY c.id, c.title;
```

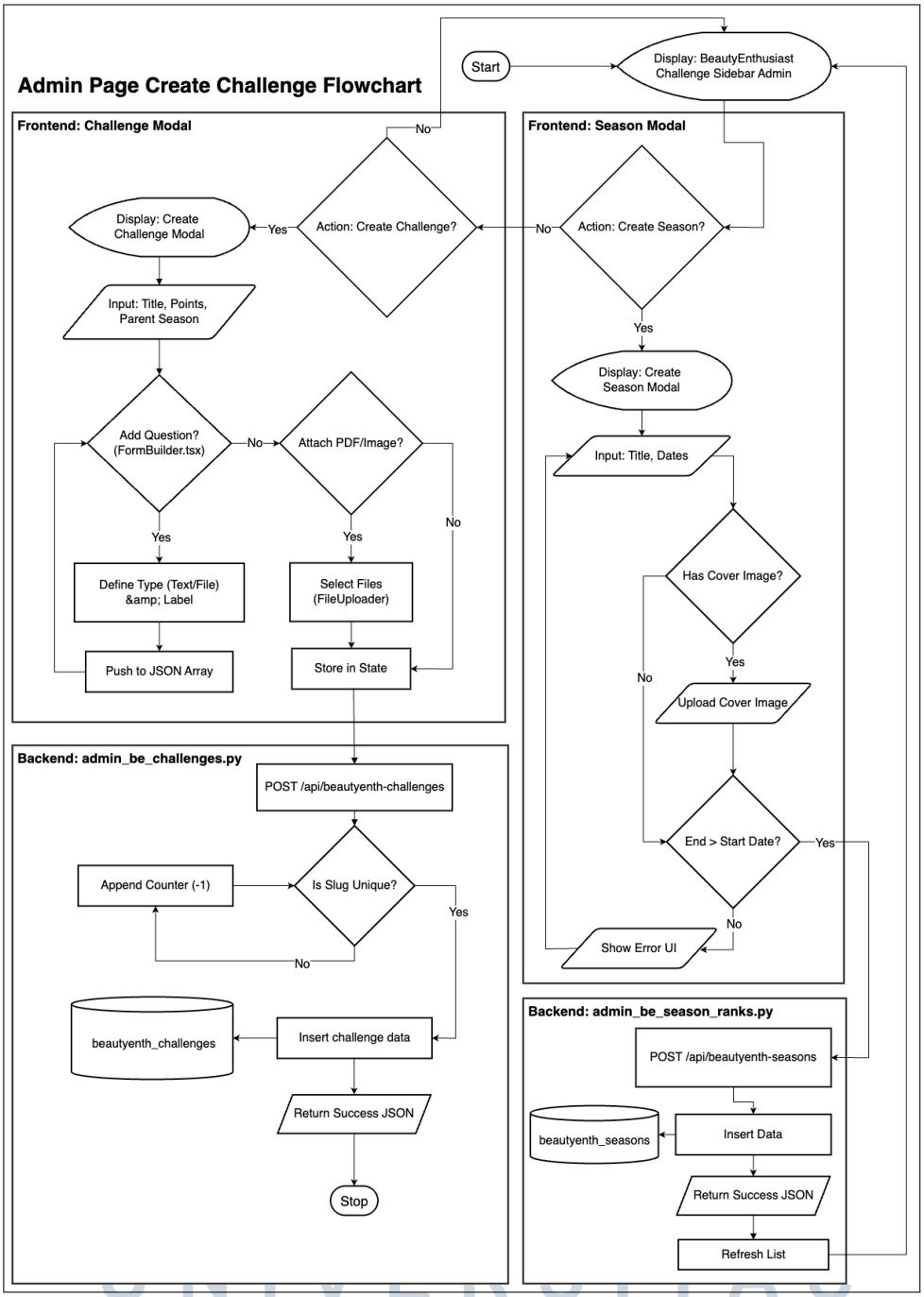
Kode 3.10: Query Menghitung Total Peserta Unik per *Challenge*

B Manajemen *Season* dan Tantangan (Sisi Administrator)

Fase inisialisasi sistem dimulai oleh administrator. Pada tahap ini, sistem memfasilitasi pembuatan siklus kompetisi (*Season*) dan penyusunan aturan main (*Challenge*). Mengingat data yang dikelola bersifat dinamis, modul ini dirancang dengan validasi berlapis untuk mencegah inkonsistensi data sebelum dipublikasikan ke pengguna.

B.1 Alur Logika Pembuatan Konten

Diagram alir pada Gambar 3.19 menjelaskan mekanisme sistem dalam memproses input administrator menjadi struktur data yang valid.



Gambar 3.19. Alur Logika *Create Season* dan *Challenge*

Berdasarkan Gambar 3.19, terdapat dua mekanisme logika penting yang dijalankan di sisi *backend*:

1. **Validasi *Temporal Season*:** Saat administrator membuat *challenge*, sistem memverifikasi bahwa *start_date* dan *end_date* tantangan berada dalam rentang

waktu *season* yang dipilih. Dari kode *admin_be_challenges.py*, *endpoint POST* melakukan validasi sebagai berikut:

```
if season_id:
    with conn.cursor() as cur:
        cur.execute(
            "SELECT start_date, end_date FROM
   beautyenth_seasons WHERE id=%s",
            (season_id,)
        )
        sr = cur.fetchone()

    if sr:
        s_start_dt = sr.get('start_date')
        s_end_dt = sr.get('end_date')

        # Validasi: challenge mulai >= season mulai
        if start_dt and start_dt < s_start_dt:
            return jsonify({'error': 'challenge_before_season
   '}), 400

        # Validasi: challenge berakhir <= season berakhir
        if end_dt and end_dt > s_end_dt:
            return jsonify({'error': 'challenge_after_season'
   }), 400

```

Kode 3.11: Validasi *Temporal Challenge* terhadap *Season Bounds*

2. **Generasi *Slug* Unik (*Recursive Logic*):** Untuk setiap tantangan yang dibuat, sistem menjalankan fungsi *slugify*. Jika judul tantangan menghasilkan URL yang sudah ada di *database*, sistem akan melakukan pengecekan berulang dengan menambahkan sufiks angka (contoh: *makeup-challenge-1*) hingga ditemukan URL yang unik. Fungsi ini dipanggil dalam endpoint:

```
provided_slug = (data.get('slug') or '').strip()
slug = unique_slug_for_challenge(provided_slug or title)

# Fungsi unique_slug_for_challenge akan:
# 1. Membuat slug dari title jika slug kosong
# 2. Mengecek apakah slug sudah ada di database
# 3. Jika ada, menambahkan suffix _1, _2, dst
# 4. Mengembalikan slug unik pertama yang ditemukan
```

```
9
```

Kode 3.12: Slug Generation pada Admin Create Challenge

B.2 Implementasi Antarmuka Pengguna

Antarmuka manajemen dibangun menggunakan *React TypeScript* dengan pendekatan *Modal-first* untuk mempercepat interaksi pengguna tanpa perlu memuat ulang halaman (*refresh*).
Dari file *Sidebar_BeautyEnth_Challeges.tsx*, administrator dapat membuat *Season* dan *Challenge* melalui modal dialog. Komponen state management menangani:

```
// Dari Sidebar_BeautyEnth_Challeges.tsx
const [showChallengeModal, setShowChallengeModal] = useState(false);
const [editing, setEditing] = useState<Challenge | null>(null);
const [form, setForm] = useState<any>({});
const [fields, setFields] = useState<FormField[]>([]);
const [previewMode, setPreviewMode] = useState(false);

// Form field builder untuk participant_form_schema
const [fields, setFields] = useState<FormField[]>([
  { id: 'field_1', label: 'Upload Foto Struk', type: 'file', required: true },
  { id: 'field_2', label: 'Esai Singkat', type: 'textarea', required: true }
]);

// Ketika submit, fields di-convert ke JSON:
const schema_str = JSON.stringify(fields);  // Ini disimpan ke beautyenth_challenges.participant_form_schema
```

Kode 3.13: State Management Admin Challenge Creation

Daftar tantangan yang telah dibuat dapat dikelola melalui antarmuka list, dengan fitur:

- Filter berdasarkan status (*upcoming*, *ongoing*, *past*)
- Filter berdasarkan *min points*
- Pencarian berdasarkan judul tantangan
- Edit/Delete tantangan yang sudah dibuat

B.3 Spesifikasi Penyimpanan Data

Perancangan basis data untuk modul ini menggunakan pendekatan *hybrid*, menggabungkan struktur *SQL* konvensional dengan tipe data *JSON* dan *BLOB* untuk fleksibilitas maksimal.
Tabel 3.9 berfungsi menyimpan *metadata* periode kompetisi:

Tabel 3.9. Struktur Tabel *beautyenth_seasons*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	CHAR(36)	*Primary Key (UUID)*
title	VARCHAR(255)	Nama *Season*
slug	VARCHAR(255)	URL unik (e.g., *winter-2024*)
image	LONGBLOB	Gambar visual season dalam format *binary*
start_date	DATETIME	Waktu mulai akses *season*
end_date	DATETIME	Waktu berakhir *season*
created_at	DATETIME	Timestamp pembuatan

Tabel 3.10 memiliki struktur yang unik karena menyimpan logika formulir di dalam kolom *database*:

Tabel 3.10. Struktur Tabel *beautyenth_challenges*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	CHAR(36)	*Primary Key (UUID)*
season_id	CHAR(36)	*Foreign Key* ke tabel *beautyenth_seasons*
slug	VARCHAR(255)	URL unik hasil generasi otomatis
title	VARCHAR(255)	Judul tantangan
description	TEXT	Penjelasan tantangan
point_reward	INT	Poin yang diberikan saat disetujui
start_date	DATETIME	Waktu mulai tantangan aktif
end_date	DATETIME	Waktu batas pengerjaan
participant_form_schema	JSON	Skema Dinamis. Menyimpan konfigurasi input (Label, Tipe, *Required*) sebagai array *JSON*
has_participant_form	TINYINT	Flag: apakah tantangan memerlukan form pengisian
need_admin_approval	TINYINT	Flag: apakah hasil perlu approval dari admin

Untuk memberikan fleksibilitas maksimal dalam konfigurasi formulir, format *JSON Schema* dirancang agar mudah diparsing dan divalidasi di sisi *backend*. Struktur ini memungkinkan *admin* untuk menambah/mengubah field tanpa proses *deployment* ulang. Berikut adalah contoh struktur *participant_form_schema* yang disimpan di basis data yang dapat dilihat pada Kode 3.14:

```
[
  {
    "id": "field_1",
    "label": "Upload Foto Struk Pembelian",
    "type": "file",
    "required": true,
    "accept": ["image/jpeg", "image/png"]
  },
  {
    "id": "field_2",
    "label": "Nama Lengkap",
```

```
    "type": "text",
    "required": true,
    "maxLength": 100
  },
  {
    "id": "field_3",
    "label": "Cerita Pengalaman (Max 500 karakter)",
    "type": "textarea",
    "required": false,
    "maxLength": 500
  }
]
```

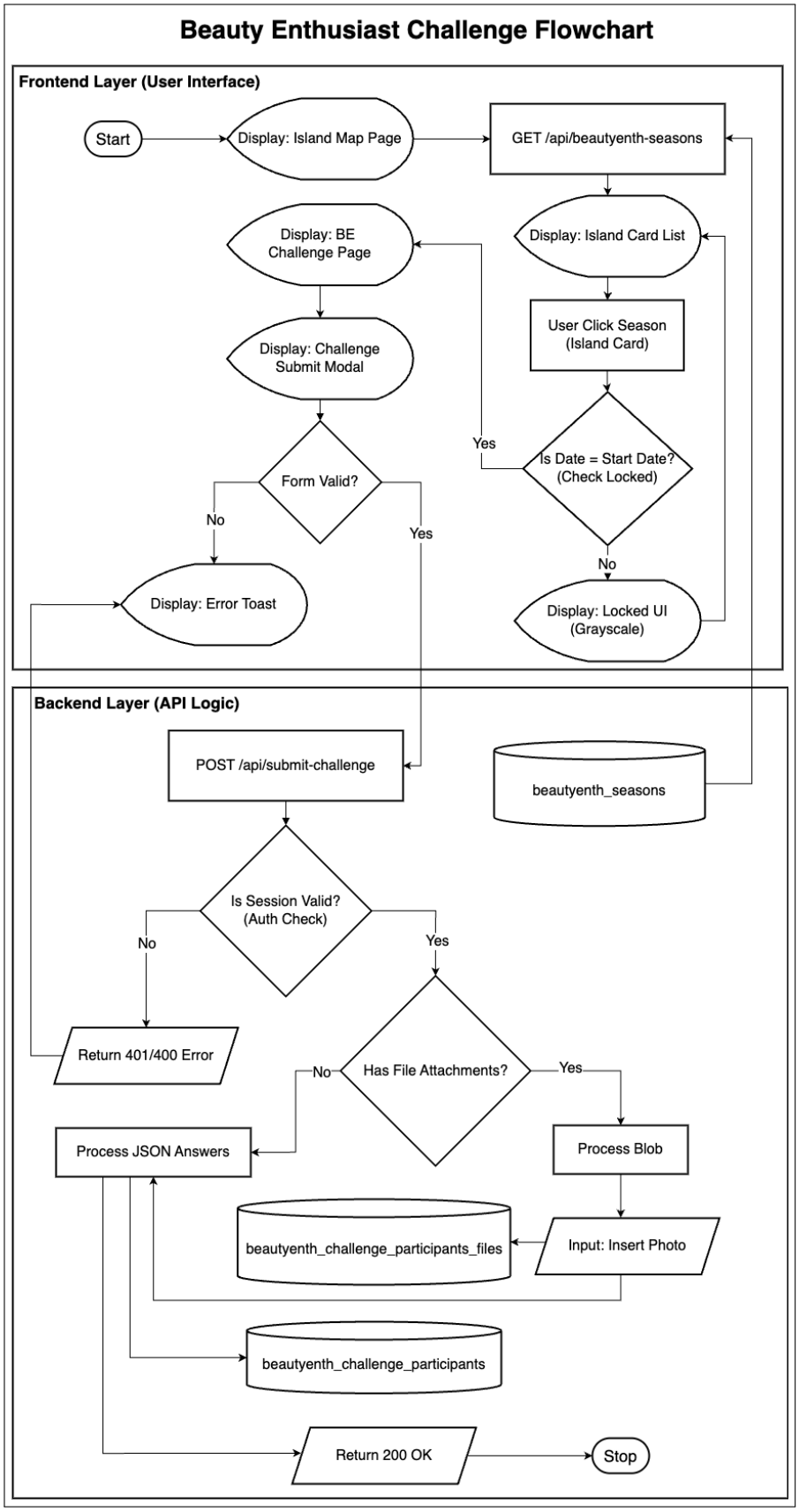
Kode 3.14: Contoh Struktur *participant_form_schema* dalam *Database*

C Eksplorasi dan Partisipasi Tantangan (Sisi Pengguna)

Setelah konten dikonfigurasi oleh administrator, pengguna dapat berinteraksi dengan sistem melalui antarmuka kalender visual. Fase ini berfokus pada pengalaman pengguna dalam menjelajahi *season* kompetisi dan melakukan pengiriman bukti *challenge* secara responsif.

C.1 Alur Logika dan Percabangan Data

Mekanisme partisipasi melibatkan sinkronisasi antara validasi di sisi antarmuka (*frontend*) dan keamanan di sisi *backend*. Diagram alur logika pada Gambar 3.20 memetakan keputusan logis yang terjadi selama proses ini.



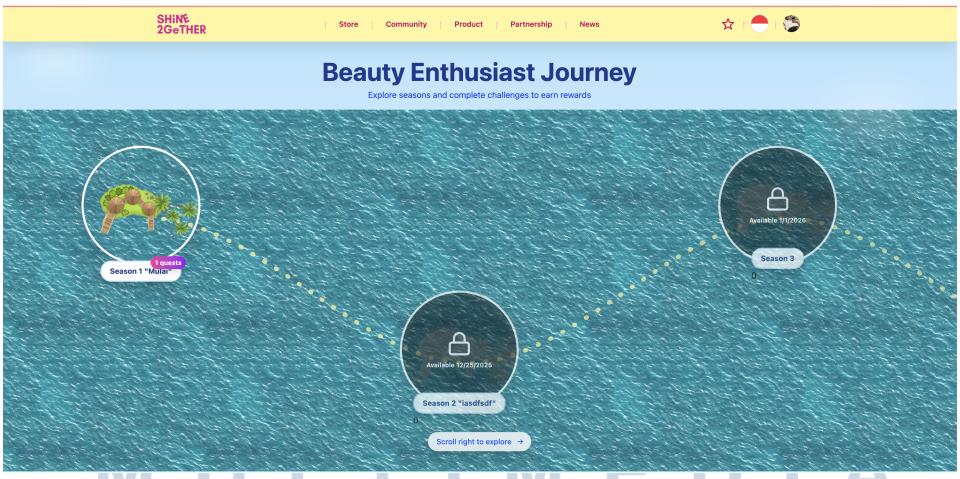
Gambar 3.20. Diagram Alur Partisipasi dengan Logika Percabangan Sistem

Berdasarkan Gambar 3.20, terdapat tiga titik keputusan krusial untuk menjaga integritas sistem:

1. **Validasi Waktu (*Client-Side*):** Sistem membandingkan waktu lokal perangkat pengguna dengan *start_date* tantangan. Jika tanggal saat ini belum mencapai waktu mulai, akses ke tantangan dikunci secara visual.
2. **Validasi Sesi (*Server-Side*):** Setiap permintaan pengiriman data (*POST*) divalidasi oleh *middleware*. Dari *be_challenges_dashboard.py*, *user* harus memiliki *cookie id* yang valid.
3. **Pemisahan *Payload* (*Server-Side*):** Sistem *backend* memilah data yang masuk berdasarkan tipenya. Data jawaban teks diproses sebagai *JSON* dan disimpan di Tabel 3.11, sedangkan lampiran berkas dipisahkan untuk disimpan di Tabel 3.12.

C.2 Implementasi Antarmuka Pengguna

Halaman utama modul ini, *BeautyEnth_Page.tsx*, me-*render* data musim dan tantangan menjadi kalender interaktif menggunakan *library react-big-calendar*. Tampilan halaman dapat dilihat pada Gambar 3.21.



Gambar 3.21. Visualisasi Kalender dengan Logika Penguncian Akses

Ketika pengguna memilih tantangan yang aktif, sistem memanggil komponen *EventDetailModal.tsx*. Modal ini mengambil data *participant_form_schema* dari *database* dan me-*render*-nya menjadi *form* interaktif dengan validasi bawaan.

C.3 Response Data dari Backend

Dari *endpoint* */api/my-be-challenges* dalam *be_challenges_dashboard.py*, *server* mengembalikan struktur data yang komprehensif mencakup informasi tantangan dan *schema* formulir. Respons ini digunakan oleh *frontend* untuk menampilkan tantangan yang sedang aktif beserta metadata lengkapnya. Berikut adalah format respons *JSON* yang diterima oleh *frontend*:

```
{
    "items": [
        {
            "participant_id": "uuid-1",
            "challenge_id": "uuid-2",
            "status": "pending",
            "answers": { "field_name": "value" },
            "challenge": {
                "id": "uuid-2",
                "slug": "beauty-challenge-1",
                "title": "Beauty Challenge #1",
                "description": "Deskripsi tantangan...",
                "point_reward": 100,
                "start_date": "2024-01-15T10:00:00",
                "end_date": "2024-01-30T23:59:59",
                "participant_form_schema": [
                    {
                        "id": "field_1",
                        "label": "Upload Foto Struk",
                        "type": "file",
                        "required": true
                    },
                    {
                        "id": "field_2",
                        "label": "Nama Lengkap",
                        "type": "text",
                        "required": true
                    }
                ]
            }
        }
    ]
}
```

Kode 3.15: Struktur Data yang Dikembalikan *Backend* untuk *Render Form*

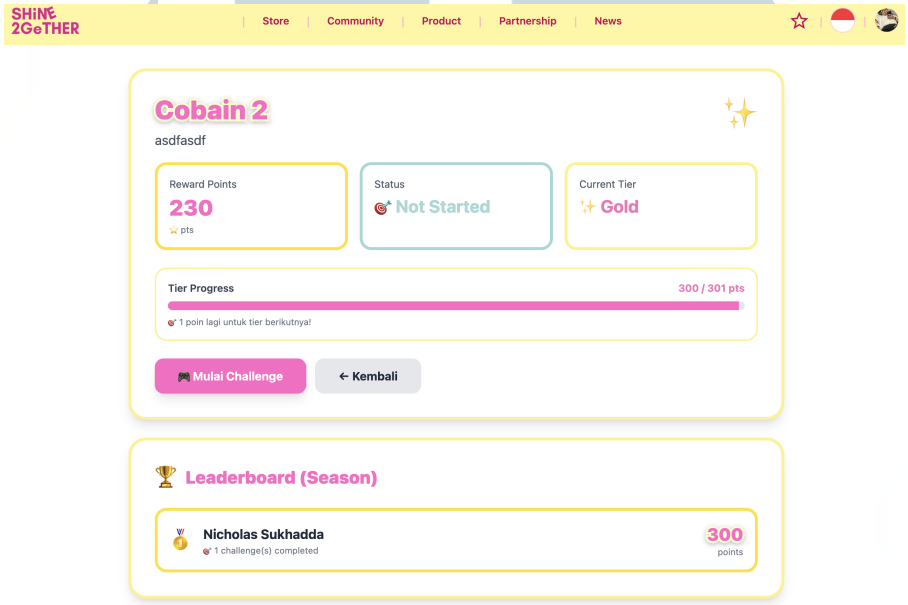
Sistem *frontend* menerjemahkan skema *JSON* yang diterima dari *backend* menjadi elemen input *HTML5* yang fungsional dan dinamis. Setiap tipe field di-*render* dengan *event handler* yang sesuai untuk menangkap input pengguna dan menyimpannya dalam state lokal sebelum dikirim kembali ke *server*. Pendekatan ini memungkinkan pengguna mengisi form dengan berbagai tipe input tanpa perlu modifikasi kode *frontend*. Implementasi *render function* dapat dilihat dalam kode berikut:

```
// Dalam EventDetailModal.tsx
const renderFormField = (field: FormField) => {
  switch(field.type) {
    case 'text':
      return (
        <input
          type="text"
          placeholder={field.label}
          value={answers[field.id] || ''}
          onChange={(e) => setAnswers({...answers, [field.id]: e.target.value})}
          required={field.required}
        />
      );
    case 'textarea':
      return (
        <textarea
          placeholder={field.label}
          value={answers[field.id] || ''}
          onChange={(e) => setAnswers({...answers, [field.id]: e.target.value})}
          required={field.required}
        />
      );
    case 'file':
      return (
        <input
          type="file"
          onChange={(e) => setFiles({...files, [field.id]: e.target.files?.[0] || null})}
          required={field.required}
        />
      );
  }
```

```
32 };
```

Kode 3.16: Render Form Dinamis dari JSON Schema

Pengguna dapat mengisi teks dan mengunggah berkas sesuai aturan yang berlaku tanpa adanya *hardcoding* pada sisi *frontend*.



Gambar 3.22. *Render* Formulir Dinamis Berdasarkan Skema *JSON*

C.4 Strategi Penyimpanan Data

Untuk menjaga performa *database* tetap optimal saat menangani ribuan partisipasi secara bersamaan, strategi penyimpanan dirancang dengan pemisahan logis berdasarkan karakteristik data. Pendekatan *normalization* ini menghindari *bottleneck* pada I/O dan memastikan query performa tetap cepat meskipun data terus bertambah sepanjang musim. Data disimpan dalam dua Tabel terpisah dengan fungsi dan karakteristik yang berbeda:

C.5 Tabel Partisipasi dan Jawaban Pengguna

Tabel 3.11 (textitbeautyenth_challenge_participants) difokuskan untuk menyimpan *metadata* partisipasi dan jawaban tekstual yang berukuran relatif ringan. Kolom-kolom penting mencakup status approval, *timestamp* pengiriman, dan jawaban dalam

format *JSON* yang terstruktur. Tabel ini sering di-query untuk mengambil daftar partisipan dan status mereka, sehingga performa read sangat penting:

Tabel 3.11. Struktur Tabel *beautyenth_challenge_participants*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	CHAR(36)	*Primary Key (UUID)*
user_id	CHAR(36)	ID pengguna (*Foreign Key* ke tabel users)
challenge_id	CHAR(36)	ID tantangan (*Foreign Key*)
answers	JSON	Jawaban teks pengguna dalam format *Key-Value*
status	ENUM	Status (*'pending'*, *'approved'*, *'rejected'*)
created_at	DATETIME	Waktu pengiriman pertama

C.6 Tabel File Lampiran Pengguna

Tabel 3.12 (*beautyenth_challenge_participant_files*) dirancang khusus untuk menyimpan data *binary* seperti gambar dan dokumen yang berukuran besar (*LONGBLOB*). Pemisahan tabel ini memastikan bahwa query pada tabel partisipasi tetap cepat tanpa beban *binary data* yang berat. Setiap file dilacak dengan *Foreign Key* ke tabel partisipasi dan nama field yang mereferensikannya dalam skema:

Tabel 3.12. Struktur Tabel *beautyenth_challenge_participant_files*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	CHAR(36)	*Primary Key (UUID)*
participant_id	CHAR(36)	*Foreign Key* ke tabel partisipasi
field_name	VARCHAR(100)	Nama field dari skema
data	LONGBLOB	Data *binary* (gambar atau dokumen)
uploaded_at	DATETIME	Timestamp upload

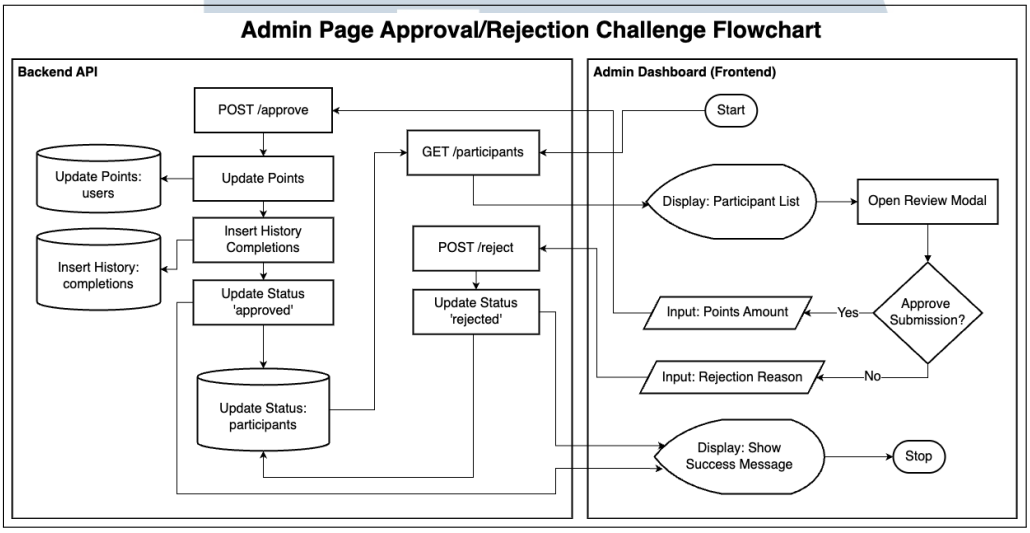
D Validasi Administrator dan Umpan Balik Pengguna

Fase terakhir dalam siklus *gamification* adalah proses moderasi dan refleksi hasil. Pada tahap ini, *administrator* memvalidasi bukti tantangan, dan sistem memberikan

umpan balik visual secara langsung kepada pengguna melalui *Dashboard*.

D.1 Alur Logika Persetujuan dan Penolakan

Mekanisme validasi dirancang menggunakan logika keputusan *binary* untuk memastikan integritas data poin. Diagram alir pada Gambar 3.23 memetakan proses keputusan tersebut dari sisi *Administrator* hingga pemrosesan di *server*.



Gambar 3.23. Diagram Alur Validasi Tantangan dengan Logika Keputusan *Biner*

Merujuk pada Gambar 3.23, sistem membagi alur eksekusi berdasarkan keputusan administrator:

- ***Reject*:** *Administrator* wajib menyertakan alasan penolakan. Sistem akan memperbarui kolom *status* pada Tabel 3.11 menjadi *'rejected'* agar pengguna dapat memperbaiki kiriman mereka.
- ***Approve*:** *Administrator* memberikan nilai poin. Sistem *backend* mengeksekusi transaksi data yang kompleks meliputi: pembaruan (*update*) status menjadi *'approved'*, pencatatan riwayat ke Tabel 3.13, dan akumulasi poin ke profil pengguna secara otomatis.

D.2 Pencatatan Riwayat Transaksi Poin

Setiap perubahan angka pada *dashboard* pengguna (seperti akumulasi poin dan perubahan peringkat) harus didasari oleh pencatatan audit trail yang presisi dan

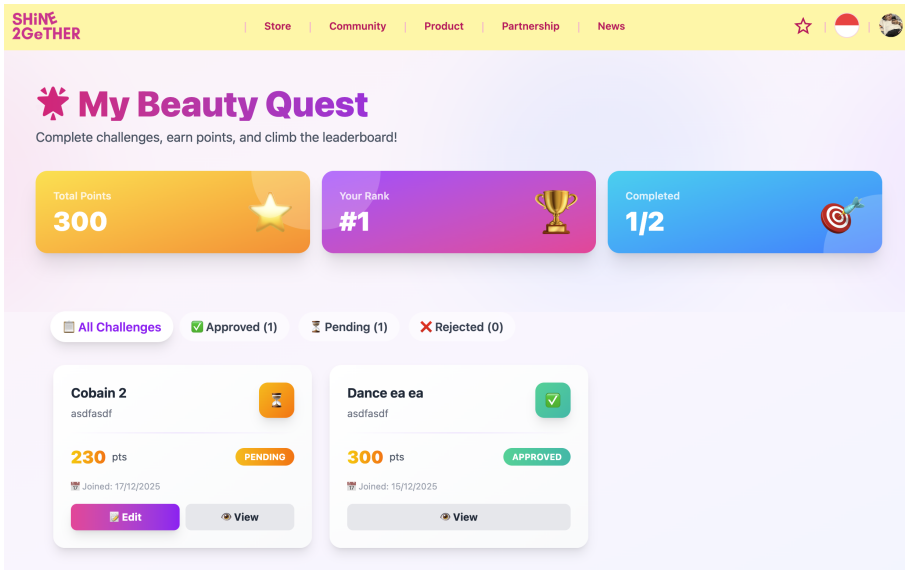
dapat dilacak untuk transparansi. Sistem menggunakan pola *append-only* untuk memastikan integritas data poin dan mencegah manipulasi saldo. Tabel 3.13 berfungsi sebagai ledger permanen yang merekam setiap pemberian poin dengan *timestamp* akurat dan referensi ke user serta tantangan:

Tabel 3.13. Struktur Tabel *beautyenth_challenges_completions*

Nama Kolom	**Tipe Data**	**Deskripsi**
id	CHAR(36)	*Primary Key (UUID)*
user_id	CHAR(36)	ID pengguna
challenge_id	CHAR(36)	ID tantangan yang diselesaikan
point_awarded	INT	Poin yang ditambahkan
completed_at	DATETIME	Waktu validasi admin

D.3 Visualisasi Hasil pada Dashboard Pengguna

Setelah *administrator* melakukan validasi, dampak dari keputusan tersebut langsung tercermin pada antarmuka pengguna. Gambar 3.24 menampilkan halaman *dashboard* yang menampilkan progres pengguna.



Gambar 3.24. Dashboard Pengguna Menampilkan Poin, Peringkat, dan Status Tantangan

Berdasarkan tampilan antarmuka pada Gambar 3.24, sistem menyajikan tiga kategori metrik kunci yang diperbarui secara *real-time* setelah *administrator* melakukan validasi. Setiap metrik diambil dari sumber data yang berbeda namun

terintegrasi dalam satu pandangan kohesif untuk memberikan gambaran lengkap tentang kemajuan pengguna dalam musim:

- ***Gamification Metrics*:** Kartu bagian atas menampilkan *Total Points* (akumulasi nilai dari Tabel 3.13) dan *Current Rank* (*ranking* di papan peringkat). Angka ini diperbarui otomatis sesaat setelah *admin* menekan tombol *"Approve"*.
- ***Submission Status*:** Daftar tantangan dilengkapi dengan indikator status visual. Label hijau bertuliskan *"APPROVED"* menandakan tantangan telah valid, sedangkan label kuning *"PENDING"* menandakan sedang dalam antrean moderasi *admin*.
- ***Challenge History*:** Pengguna dapat melihat riwayat tantangan yang telah diikuti beserta poin yang diraih. Informasi ini diambil dari Tabel 3.13 melalui *endpoint /api/my-be-challenges*.

3.3.4 Projek Magang (SHiNE2GeTHER Stock WebApp)

Sistem manajemen stok *S2G Stock WebApp* adalah *website* yang mengelola aliran barang dari pusat distribusi hingga ke lokasi akhir penjualan dan distribusi. Sistem ini dirancang untuk melacak setiap item barang dengan detail, memastikan ketersediaan stok yang optimal, dan mengotomatisasi proses persetujuan untuk transaksi strategis.

A Cakupan dan Komponen Utama

Manajemen stok dalam konteks aplikasi ini mencakup seluruh siklus hidup barang mulai dari penerimaan *batch* barang di pusat distribusi, kategorisasi stok berdasarkan kondisi, distribusi ke berbagai lokasi regional dan *retail*, hingga pengelolaan stok untuk keperluan kantor dan platform *e-commerce*.
Sistem manajemen stok terdiri dari beberapa komponen utama yang saling terhubung:

1. *Batch* Penerimaan (*Stock Batch*): Kumpulan barang yang diterima dalam satu waktu dengan kode unik untuk melacak asal dan tanggal penerimaan
2. *Item Stok* (*Stock Item*): Unit individual barang dengan kategori dan sub-kategori spesifik sesuai kondisi dan tujuan penggunaan

3. Lokasi Penyimpanan (*Location*): Tempat fisik menyimpan barang dengan kode unik dan hierarki *parent-child* untuk struktur organisasi

4. Kategori Stok (*Stock Category*): Klasifikasi barang berdasarkan kondisi fisik dan kelayakan jual (*BSK*, *E*, *BR*)

5. Pesanan Gudang (*Warehouse Order*): Transaksi distribusi stok antar lokasi yang memerlukan pelacakan dan persetujuan

6. Pengiriman (*Shipment*): Proses pengiriman barang dengan pelacakan detail, kemasan, dan konfirmasi penerimaan di lokasi tujuan

B Kamus Kode Lokasi dan Kategori

Untuk memudahkan pemahaman tentang sistem manajemen stok, penting untuk mengenal kode-kode yang digunakan dalam setiap lokasi dan kategori barang. Berikut dijelaskan definisi dan penjelasan lengkap untuk setiap kode yang berlaku dalam sistem *S2G Stock WebApp*.

B.1 Definisi Kode Lokasi

Setiap lokasi dalam sistem *S2G Stock WebApp* memiliki kode unik yang merepresentasikan lokasi fisik dan fungsinya dalam jaringan distribusi. Kode lokasi mengikuti struktur hierarki yang memungkinkan pelacakan alur distribusi dari pusat manufaktur hingga lokasi akhir.

B.1.1 Tabel Kode Lokasi

Berikut adalah penjelasan kode-kode lokasi utama dalam sistem yang ada pada Tabel 3.14.

Tabel 3.14. Kode lokasi dalam sistem *S2G Stock WebApp*

Kode Lokasi	**Deskripsi**	***Parent***
FAB/CH	Pabrik/Manufaktur Utama (*Manufacturing* Pusat Produksi)	-
GMA	Gudang Master/Utama (*Distribution Center* Sekunder)	*FAB/CH*
GDB	Gudang Distribusi *Backup* (*Hub* Distribusi regional)	*GMA*
GOF	Gudang *Office*/Kantor (Pusat *Retail* dan Kantor)	*GMA*
DBW/JKT	Distributor Jawa (Distributor regional - Tujuan Akhir)	*GDB*
OFCL/ECOM	*Office E-Commerce* (Pusat Kontrol *E-Commerce*)	-
ECOM/TXT	*E-Commerce Tokopedia* (*Fulfillment Center*)	*OFCL/ECOM*
ECOM/SH	*E-Commerce Shopee* (*Fulfillment Center*)	*OFCL/ECOM*
OFCL/CH	*Office* Utama (Pusat Administrasi Kantor)	-

B.2 Definisi Kategori Stok

Setiap *item stok* diklasifikasikan ke dalam kategori berdasarkan kondisi fisik dan kelayakan untuk dijual. Kategori ini membantu dalam pengelolaan stok yang efisien dan memastikan hanya produk berkualitas yang sampai ke lokasi tujuan distribusi.

B.2.1 Tabel Kategori Stok

Berikut adalah kategori stok utama dalam sistem pada Tabel 3.15:

Tabel 3.15. Kategori stok dalam sistem *S2G Stock WebApp*

Kode Kategori	**Deskripsi**
BSK	Barang Sempurna/Baru (Produk dalam kondisi terbaik, siap jual ke pelanggan atau distributor)
E	*Emergency*/Darurat (Barang untuk memenuhi permintaan mendesak, dapat digunakan jika *BSK* tidak tersedia)
BR	Barang Rusak (Produk yang tidak sesuai standar kualitas, perlu evaluasi lebih lanjut)

B.3 Definisi Sub-Kategori Stok

Sub-kategori lebih detail mengklasifikasikan *item* berdasarkan jenis dan bentuk pengiriman barang dalam sistem distribusi. Penggunaan sub-kategori memastikan setiap *item stok* dapat diidentifikasi dengan spesifik sesuai dengan format dan tujuan penggunaannya.

B.3.1 Tabel Sub-Kategori Stok

Berikut adalah sub-kategori stok yang tersedia dalam sistem pada Tabel 3.16.

Tabel 3.16. Sub-kategori stok dalam sistem *S2G Stock WebApp*

Kode Sub-Kategori	**Deskripsi**
FN	*Final*/Normal (Produk *final* dalam kemasan standar, siap distribusi)
BT	Botol (Botol Produk kemasan)
BX	*Box*/Kemasan (Produk dalam kemasan khusus atau *bundling* untuk tujuan tertentu)

C Struktur Hierarki dan Alur Distribusi Stok

Sistem manajemen stok *S2G Stock WebApp* dirancang dengan struktur hierarki yang jelas untuk memastikan alur distribusi barang berjalan secara terorganisir dari pusat manufaktur hingga ke tujuan akhir. Setiap lokasi dalam jaringan memiliki peran

spesifik yang saling terhubung membentuk rantai distribusi yang efisien. Berikut dijelaskan secara detail tentang struktur hierarki dan alur distribusi untuk setiap jalur utama.

C.1 Overview Jaringan Distribusi

Jaringan distribusi *S2G Stock WebApp* mengikuti model *hirarkis* dengan aliran stok dari pusat manufaktur ke lokasi-lokasi distribusi regional, kantor retail, dan platform *e-commerce*. Struktur ini dirancang untuk efisiensi logistik dan kontrol stok yang ketat dengan tujuan akhir yang jelas untuk setiap jalur distribusi.

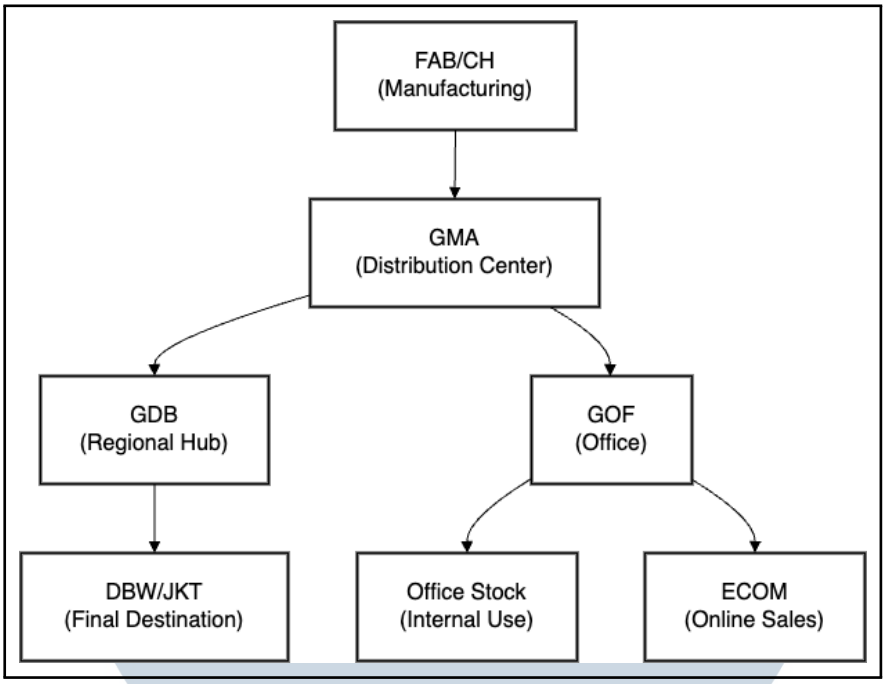
C.1.1 Tujuan Akhir Setiap Jalur Distribusi

Sistem memiliki tiga jalur distribusi utama dengan tujuan akhir yang berbeda:

- Jalur Distribusi regional: Produk mengalir dari pusat manufaktur menuju distributor regional (contoh: *DBW/JKT*) yang menjadi tujuan akhir untuk penjualan ke pelanggan lokal
- Jalur Kantor dan *Retail*: Produk disimpan di *GOF* sebagai stok kantor untuk penggunaan operasional internal dan penjualan *retail* langsung
- Jalur *E-Commerce*: Produk dialokasikan ke *platform e-commerce* (*Tokopedia, Shopee*) melalui pusat kontrol terpusat untuk penjualan *online*

C.2 Diagram Alur Distribusi Primer

Berikut adalah visualisasi alur distribusi stok dari pusat manufaktur hingga ke lokasi akhir, seperti yang ditampilkan pada Gambar 3.25.



Gambar 3.25. Struktur hierarki jaringan distribusi

Diagram ini menunjukkan bagaimana stok mengalir secara hierarki dari pusat manufaktur (*FAB/CH*) ke pusat distribusi utama (*GMA*), kemudian bercabang ke hub regional (*GDB*) dan kantor (*GOF*). Dari *GDB*, stok didistribusikan ke distributor lokal seperti *DBW/JKT*, sementara dari *GOF*, stok dialokasikan untuk kebutuhan internal kantor dan platform *e-commerce*.

C.3 Cabang Distribusi Regional

Cabang distribusi regional merupakan jalur utama yang menangani distribusi stok ke seluruh wilayah melalui *hub regional*. Jalur ini memastikan produk tersebar secara merata ke setiap distributor regional dengan kontrol kualitas dan *monitoring* yang ketat. Berikut dijelaskan secara rinci tentang alur operasional dan karakteristik cabang distribusi regional.

C.3.1 Alur Distribusi Regional

Alur distribusi regional mengikuti rute: *FAB/CH* $\rightarrow$ *GMA* $\rightarrow$ *GDB* $\rightarrow$ *DBW/JKT* (*Final Destination*)

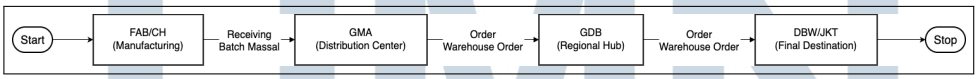
Cabang distribusi regional memiliki peran sebagai *hub regional* yang menerima stok dalam volume besar dari pusat distribusi dan mendistribusikannya ke distributor

wilayah. Lokasi distributor seperti *DBW/JKT* merupakan tujuan akhir jalur distribusi regional, di mana distributor bertugas menjual produk ke pelanggan akhir di wilayah masing-masing. Berikut Karakteristik alur distribusi regional:

- Menerima stok dalam jumlah besar dari pusat distribusi (*GMA*) melalui *batch* massal
- Menyimpan stok untuk kebutuhan regional jangka menengah sebagai *buffer inventaris*
- Mendistribusikan stok ke distributor lokal (*DBW/JKT* dan lokasi regional lainnya) melalui pesanan gudang
- Mengelola kategori stok (*BSK*, *E*, *BR*) dengan prioritas pada kategori *BSK* untuk distributor
- *Monitoring* stok untuk memastikan ketersediaan produk di setiap distributor regional

C.3.2 Contoh Alur Spesifik Distributor Jawa

Berikut adalah alur transaksi khusus untuk lokasi distributor Jawa yang menunjukkan proses detail dari penerimaan *batch* hingga sampai di tujuan akhir. Alur ini ditampilkan pada Gambar 3.26.



Gambar 3.26. Alur transaksi khusus distributor Jawa dari penerimaan*batch* di manufaktur hingga tujuan akhir di DBW/JKT

Alur ini menunjukkan bahwa stok mengalir dari pusat manufaktur (*FAB/CH*) dengan penerimaan *batch* massal di pusat distribusi utama (*GMA*), kemudian didistribusikan ke *hub* regional (*GDB*) melalui pesanan gudang (*Warehouse Order*), dan akhirnya sampai ke distributor Jawa (*DBW/JKT*) sebagai tujuan akhir untuk penjualan ke pelanggan lokal.

C.4 Cabang Kantor dan Retail

Cabang kantor dan *retail* merupakan jalur distribusi khusus yang menangani stok untuk keperluan operasional kantor dan penjualan *retail* langsung. Jalur

ini memiliki karakteristik unik karena melayani dua tujuan sekaligus: kebutuhan internal kantor dan alokasi stok untuk platform *e-commerce*. Berikut dijelaskan alur operasional dan fungsi cabang kantor dan *retail* secara komprehensif.

C.4.1 Alur Kantor dan Retail

Alur kantor dan *retail* mengikuti rute: *FAB/CH* $\rightarrow$ *GMA* $\rightarrow$ *GOF* $\rightarrow$ Stok Kantor dan *E-Commerce*
Cabang kantor dan *retail* (*GOF*) berfungsi sebagai pusat pengumpulan stok untuk dua tujuan utama: (1) kebutuhan kantor internal dan *retail* penjualan langsung, dan (2) alokasi stok untuk operasi *e-commerce*. Berbeda dengan jalur distribusi regional, *GOF* tidak mengirimkan stok ke distributor eksternal, melainkan mengelola stok untuk penggunaan internal dan platform *online*.
Karakteristik alur kantor dan *retail*:

- Menerima stok dari *GMA* dengan volume lebih kecil dibanding alur distribusi regional
- Fokus pada kategori *BSK* (Barang Sempurna) untuk menjaga kualitas produk yang dijual
- Mengelola stok *segment* khusus: *office* (*OFCL*), *community* (*CMTY*), *partnership* (*PRTN*)
- Menyimpan stok untuk kebutuhan operasional kantor (*perlengkapan*, *inventaris* internal)
- Melayani pesanan *e-commerce* dengan alokasi stok dari *inventaris* yang tersedia

C.5 Cabang E-Commerce

Cabang *e-commerce* merupakan jalur distribusi khusus yang menangani stok untuk platform penjualan *online*. Jalur ini memiliki struktur terpusat dengan pusat kontrol dan beberapa *fulfillment center* yang didistribusikan per platform. Sistem ini dirancang untuk memastikan efisiensi dalam alokasi stok dan memenuhi permintaan penjualan *online* yang dinamis. Berikut dijelaskan alur operasional dan mekanisme *approval* untuk cabang *e-commerce*.

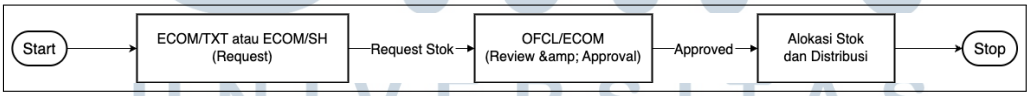
C.5.1 Alur E-Commerce

Alur *e-commerce* mengikuti rute: *FAB/CH* $\rightarrow$ *GMA* $\rightarrow$ *OFCL/ECOM* $\rightarrow$ *ECOM/TXT*, *ECOM/SH*
Cabang *e-commerce* mengelola stok khusus untuk platform *e-commerce* dengan struktur pusat kontrol terpusat dan *fulfillment center* per platform. Alur ini memastikan stok dialokasikan secara terkoordinasi dan transparan dengan kontrol kualitas yang ketat. Struktur dan karakteristik:

- *OFCL/ECOM*: Pusat kontrol dan *approval* terpusat untuk semua *request* stok *e-commerce*
- *ECOM/TXT*: *Fulfillment center* untuk platform *Tokopedia* dengan manajemen stok independen
- *ECOM/SH*: *Fulfillment center* untuk platform *Shopee* dengan manajemen stok independen
- Setiap pusat *e-commerce* menerima alokasi stok dari *GMA* atau *OFCL/ECOM* berdasarkan *request* dan *approval*
- Proses *request* dan *approval* untuk stok *e-commerce* dikelola tersentralisasi untuk efisiensi dan kontrol

C.5.2 Alur Request dan Approval Stok E-Commerce

Berikut adalah alur *request* dan *approval* stok khusus untuk cabang *e-commerce* yang menunjukkan mekanisme persetujuan dan alokasi stok pada Gambar 3.27.



Gambar 3.27. Alur *request* dan *approval* stok untuk cabang *e-commerce*

Proses ini memastikan setiap *request* stok dari platform *e-commerce* (ECOM/TXT atau ECOM/SH) diproses melalui pusat kontrol terpusat (*OFCL/ECOM*) untuk mendapatkan persetujuan, sebelum akhirnya stok dialokasikan dan didistribusikan ke platform masing-masing.

D Alur Lengkap Contoh Transaksi

Untuk memberikan gambaran lengkap tentang bagaimana sistem bekerja *end-to-end*, berikut adalah alur transaksi dari penerimaan *batch* hingga distribusi *final* ke tujuan akhir. Pemahaman tentang alur lengkap ini penting untuk mengetahui setiap tahap proses dan kontrol kualitas yang diterapkan di setiap fase distribusi.

1. Penerimaan Batch: Batch barang diterima di *GMA* dari *FAB/CH* dengan kode *batch* unik (contoh: B00125/251208) dan dicatat dalam sistem
2. Kategori Stok: Batch dievaluasi dan didistribusikan ke kategori *BSK*, *E*, atau *BR* berdasarkan kondisi fisik produk
3. *Item* Stok: Setiap kategori dibagi menjadi *item* stok individual dengan kode detail yang mencakup lokasi, kategori, dan identifikasi produk (contoh: BSK-MGS-EI2Y10M)
4. Pesanan Gudang: Lokasi lain (*GDB*, *GOF*, atau *ECOM*) mengirim pesanan untuk stok yang mereka butuhkan melalui sistem pesanan gudang
5. Pengiriman: Pesanan dikemas ke dalam *shipment* dan dikirim dengan *tracking code* unik untuk pelacakan *real-time*
6. Penerimaan: Lokasi tujuan menerima *shipment*, melakukan verifikasi, dan membuat *batch* baru dari stok yang diterima untuk *inventaris* lokal
7. *Distribusi* Akhir: Stok siap untuk tujuan akhir sesuai jalur distribusi: distributor untuk penjualan regional, stok kantor untuk operasional, atau *e-commerce* untuk penjualan *online*

E Ringkasan

Sistem manajemen stok *S2G Stock WebApp* mengintegrasikan kompleksitas distribusi *multi-lokasi* dengan kontrol *granular* atas setiap unit barang. Melalui struktur hierarki yang jelas dan proses yang terstandar, sistem memastikan efisiensi dalam pengelolaan *inventaris* sambil tetap menjaga *transparency* dan *accountability* di setiap tahap alur distribusi. Dengan fokus pada *monitoring* stok di semua wilayah distributor dan kantor, sistem mendukung pengambilan keputusan yang lebih baik untuk optimasi *inventaris* dan peningkatan kepuasan pelanggan.

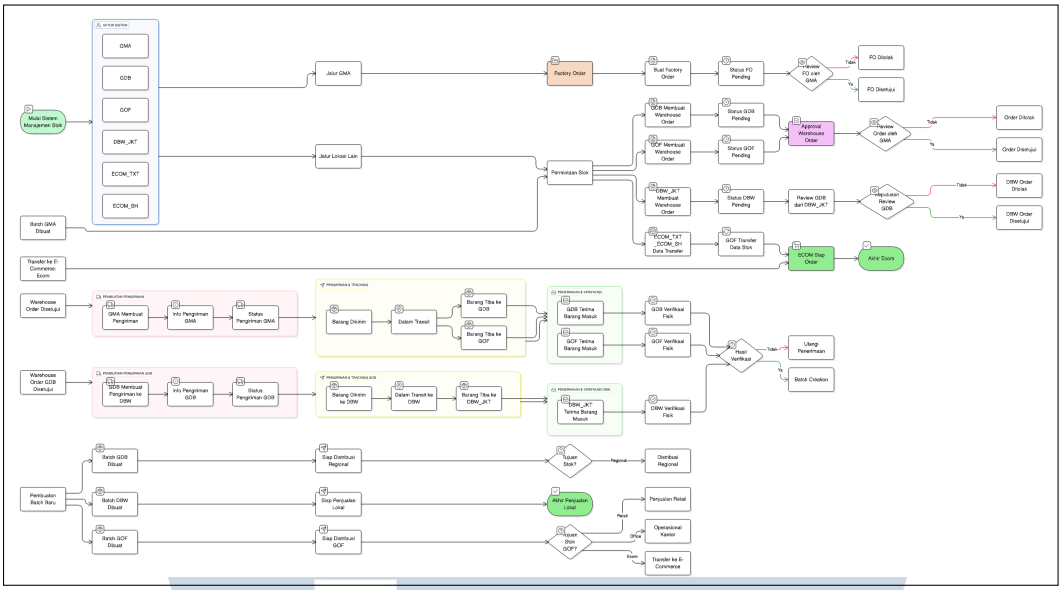
F Flowchart Sistem Manajemen Stok

Untuk memvisualisasikan alur lengkap sistem manajemen stok *S2G Stock WebApp* secara komprehensif, berikut adalah *flowchart* yang menggambarkan seluruh proses dari penerimaan *batch* hingga distribusi *final* ke berbagai tujuan akhir. *Flowchart* ini dirancang untuk menunjukkan bagaimana setiap lokasi dalam jaringan distribusi berinteraksi dan mengalirkan stok melalui berbagai tahapan proses.

F.1 Struktur Flowchart

Flowchart sistem manajemen stok terdiri dari tujuh tahapan utama yang merepresentasikan setiap fase dalam siklus distribusi:

1. *Stage 0: Factory Order* - Khusus untuk GMA yang menerima pesanan dari manufaktur
2. *Stage 1: Request* - Lokasi *child* mengirimkan pesanan gudang ke lokasi parent
3. *Stage 2: Approval* - Lokasi *parent* me-*review* dan menyetujui pesanan
4. *Stage 3: Shipment Creation* - Pembuatan *shipment* dengan kode *tracking* dan informasi kurir
5. *Stage 4: Delivery & Tracking* - Barang dikirim dengan pelacakan *real-time*
6. *Stage 5: Receiving & Verification* - Lokasi tujuan menerima dan memverifikasi barang
7. *Stage 6: Batch Creation* - Sistem otomatis membuat *batch* baru di lokasi tujuan
8. *Stage 7: Distribution Ready* - *Batch* siap untuk tujuan akhir sesuai jalur distribusi



Gambar 3.28. *Flowchart* lengkap sistem manajemen stok *S2G Stock WebApp*

F.2 Alur Utama Flowchart

Flowchart yang ditampilkan pada Gambar 3.28 mencakup semua jalur distribusi utama dalam sistem:

F.2.1 Jalur 1: Factory Order untuk GMA

Jalur ini khusus untuk *GMA* yang menerima pesanan dari pusat manufaktur (*FAB/CH*):

- *GMA* membuat *Factory Order* dari manufaktur dengan informasi jumlah unit dan tanggal pengiriman
- Pesanan di-*review* dan di-*approve* untuk memastikan kualitas dan ketersediaan ruang penyimpanan
- Setelah *approved*, *batch* barang langsung tersedia di *inventory GMA* untuk didistribusikan ke lokasi lain
- *Rejected order* masuk ke status *Order* Ditolak dan tidak dilanjutkan ke tahap berikutnya

F.2.2 Jalur 2: Warehouse Order untuk GDB dan GOF dari GMA

Jalur ini menangani pesanan gudang dari *GDB* dan *GOF* ke lokasi *parent GMA*:

- *GDB* dan *GOF* membuat *Warehouse Order* dengan status *PENDING* untuk meminta stok
- *GMA* me-*review order* dari kedua lokasi berdasarkan ketersediaan stok
- Jika *approved*, *GMA* membuat *shipment* dengan *KF Code* dan informasi kurir
- Barang dikirim dengan status *ON_SHIPPING* dan dapat dilacak secara *real-time*
- Lokasi tujuan menerima, memverifikasi, dan sistem membuat *batch* baru
- *Batch* siap untuk distribusi sesuai dengan tujuan akhir masing-masing lokasi

F.2.3 Jalur 3: Warehouse Order untuk DBW/JKT dari GDB

Jalur ini menangani pesanan gudang dari distributor Jawa *(DBW/JKT)* ke *parent GDB*:

- *DBW/JKT* membuat *Warehouse Order* ke *GDB* sebagai *parent location*
- *GDB* me-*review order* dan melakukan *approval* jika stok tersedia
- *GDB* membuat *shipment* dengan *tracking code* untuk pengiriman ke *DBW/JKT*
- Barang dalam transit dengan status *ON_SHIPPING*
- *DBW/JKT* menerima, memverifikasi, dan membuat *batch* baru untuk *inventory* lokal
- *Batch* di *DBW/JKT* siap untuk penjualan lokal kepada *end-customer*

F.2.4 Jalur 4: E-Commerce Stock Transfer dari GOF

Jalur ini khusus untuk alokasi stok ke platform *e-commerce* tanpa *shipment* fisik:

- *ECOM/TXT* dan *ECOM/SH* membuat *request* stok transfer dari *GOF*

– *GOF* melakukan transfer data stok secara langsung (tanpa *shipment* fisik)

– Transfer adalah proses data yang *instant* - tidak ada proses pengiriman barang

– *Batch* baru otomatis dibuat di *ECOM/TXT* dan *ECOM/SH* untuk *inventory online*

– Stok siap untuk proses pesanan *e-commerce* dari pelanggan

F.3 Decision Points dalam Flowchart

Flowchart mencakup beberapa *decision points* kritis yang menentukan alur proses:

– *Parent Review*: *GMA* atau *GDB* me-*review order* berdasarkan ketersediaan stok (*Approve/Reject*)

– *Verification Result*: Lokasi tujuan memverifikasi barang yang diterima (*Ya/Tidak*)

– *Distribution Destination*: Setelah *batch created*, ditentukan tujuan stok (*Regional/Retail/E-Commerce*)

F.4 Status dan End Points

Setiap alur dalam *flowchart* memiliki *end points* yang menunjukkan status *final*:

– *Ready for Distribution*: *Batch* di *GDB* siap untuk dijual ke distributor regional

– *Ready for Retail Sales*: *Batch* di *GOF* siap untuk penjualan *retail* dan operasional kantor

– *Ready for Online*: *Batch* di *ECOM* siap untuk *order e-commerce*

– *Ready for Local Sales*: *Batch* di *DBW/JKT* siap untuk penjualan lokal

– *Order* Ditolak: Pesanan ditolak karena stok *insufficient* atau alasan lainnya

F.5 Integrasi Multi-Lokasi

Flowchart menunjukkan bagaimana sistem mengintegrasikan operasi *multi-lokasi*:

- Setiap lokasi dapat bertindak sebagai *parent* atau *child* sesuai hierarki (*GMA* adalah *parent* untuk *GDB/GOF*, *GDB* adalah *parent* untuk *DBW/JKT*)
- Proses *request-approval-shipment-receive* diulang di setiap level untuk memastikan kontrol kualitas
- *Batch creation* otomatis memastikan *inventory* di setiap lokasi selalu tersinkronisasi dengan sistem
- *E-commerce* memiliki alur khusus tanpa *shipment* fisik untuk efisiensi dan kecepatan alokasi stok

F.6 Keuntungan Visualisasi Flowchart

Flowchart ini memberikan beberapa keuntungan dalam pemahaman sistem:

- *Clarity*: Visualisasi yang jelas tentang setiap tahapan proses dan *decision points*
- *Completeness*: Mencakup semua jalur distribusi dari manufaktur hingga *end-user*
- *Traceability*: Memudahkan pelacakan alur barang dari sumber hingga tujuan akhir
- *Process Understanding*: Membantu semua *stakeholder* memahami *workflow* sistem secara menyeluruh
- *Training Reference*: Dapat digunakan sebagai referensi untuk *training* pengguna baru

G Factory Order (Pesanan Manufaktur)

Factory Order adalah transaksi pesanan yang dibuat oleh *GMA* (Gudang Master/Utama) kepada *FAB* (*Pabrik*/Manufaktur) untuk melakukan produksi *batch* barang baru sesuai kebutuhan *inventaris*. Sistem ini dirancang dengan mekanisme *approval* terpusat karena pihak *FAB* (*Pabrik*) tidak memiliki akses langsung

ke *website* stok, sehingga seluruh proses pemesanan dan persetujuan dikelola sepenuhnya oleh *GMA* sebagai pusat distribusi utama.

G.1 Peran GMA dalam Factory Order

GMA memiliki peran ganda dalam sistem *Factory Order*: sebagai *requestor* (peminta pesanan) dan sebagai *approver* (penyetuju pesanan). Peran ganda ini diperlukan karena struktur organisasi di mana *FAB* tidak terlibat langsung dalam sistem *website* stok. Berikut dijelaskan alasan dan mekanisme peran *GMA*:

G.1.1 Alasan GMA Menjadi Requestor dan Approver

Beberapa alasan strategis mengapa *GMA* menangani kedua peran tersebut:

- *FAB* Tidak Memiliki Akses *Website*: Pihak *FAB* (Pabrik) sebagai produsen tidak memiliki akses ke sistem *website stock management*, sehingga tidak dapat membuat atau menyetujui pesanan secara langsung
- Kebutuhan *Inventaris* Terpusat: *GMA* sebagai pusat distribusi utama memiliki *visibilitas* penuh terhadap kebutuhan stok di seluruh jaringan distribusi, sehingga dapat mengambil keputusan pemesanan yang optimal
- Kontrol Kualitas dan Kuantitas: *GMA* dapat memverifikasi kuantitas yang diminta, kualitas spesifikasi produk, dan validasi *batch* sebelum memberikan *approval* untuk produksi di *FAB*
- *Efisiensi Operasional*: Dengan *GMA* sebagai *single point of contact*, proses *approval* menjadi lebih efisien tanpa perlu koordinasi *multi-stakeholder* dengan pihak *FAB* yang tidak *online*
- *Audit Trail* dan *Accountability*: Semua transaksi tercatat dengan jelas melalui satu akun *GMA*, memudahkan *tracking* dan akuntabilitas untuk keperluan *audit*

G.2 Proses Pembuatan Factory Order

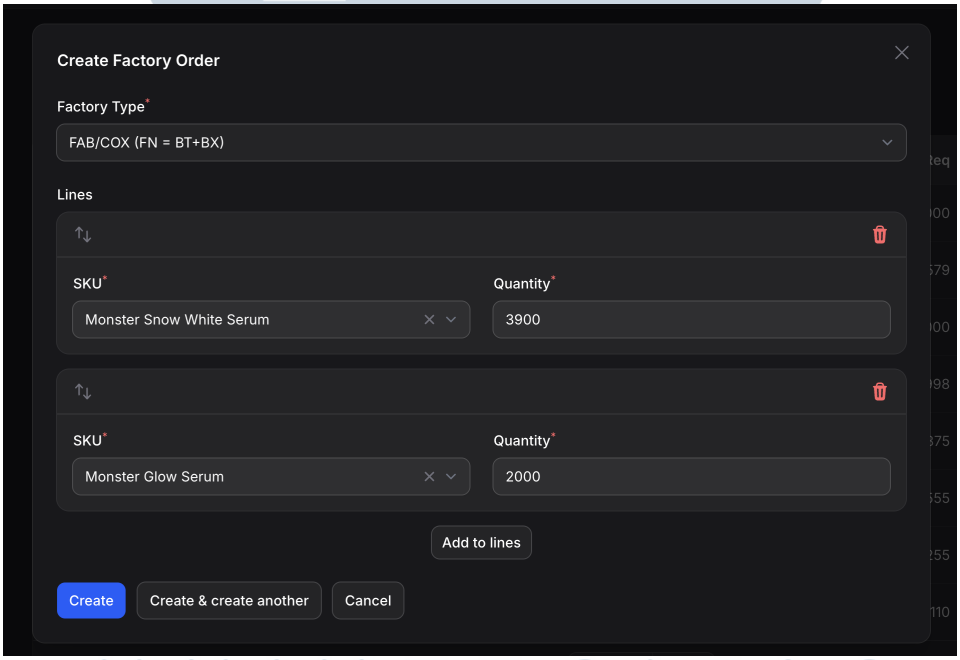
Proses pembuatan *Factory Order* dimulai dari *GMA* yang mengidentifikasi kebutuhan produksi baru berdasarkan analisis *inventaris* dan permintaan dari distributor atau cabang lainnya.

G.2.1 Langkah 1: Akses Modul Create Factory Order

GMA membuka modul *create Factory Order* di sistem. Pada tampilan *form*, *GMA* diminta untuk memilih tipe pabrik yang menentukan jenis *batch* dan sub-kategori produk yang akan diproduksi. Contoh yang tersedia adalah:

– *FAB/COX* (*FN = BT+BX*): Pesanan *batch* dari pabrik dengan sub-kategori kombinasi Botol (*BT*) dan *box*/kemasan (*BX*), yang akan dikemas dalam format *final* (*Final/Normal*)

Pemilihan tipe pabrik ini penting karena menentukan spesifikasi produksi dan format pengiriman dari *FAB* ke *GMA*. *Form* ini juga menampilkan *section lines* di mana *GMA* dapat menambahkan *item-item* yang akan diproduksi. Contoh tampilan dalam mengakses modul *create Factory Order* dapat dilihat pada Gambar3.29.



Gambar 3.29. *Form create Factory Order* dengan pemilihan tipe pabrik dan item produksi

G.2.2 Langkah 2: Menambahkan Item Produksi

Setelah memilih tipe pabrik, *GMA* menambahkan *item-item* produk yang akan diproduksi dengan mengklik tombol *add to lines*. Untuk setiap item, *GMA* harus mengisi:

– *SKU* (*Stock Keeping Unit*): Identitas unik produk yang akan diproduksi (contoh: *Monster Snow White Serum* dengan kode *MSWS*, *Monster Glow Serum* dengan kode *MGS*)

– *Quantity*: Jumlah unit yang diminta untuk diproduksi (contoh: 3900 unit untuk *MSWS*, 2000 unit untuk *MGS*)

Setiap *item* ditampilkan dalam satu baris dengan opsi untuk menghapus atau mengubah urutan melalui *icon drag-and-drop*. *GMA* dapat menambahkan *multiple item* dalam satu pesanan *Factory Order* untuk efisiensi. Tombol *add to lines* memungkinkan penambahan *item* tambahan dengan mudah.

G.2.3 Langkah 3: Proses Pembuatan Pesanan

Setelah semua item ditambahkan, *GMA* memiliki tiga opsi aksi:

1. *Create*: Membuat pesanan *Factory Order* dan langsung menyimpannya ke sistem (tombol berwarna biru)
2. *Create & create another*: Membuat pesanan *Factory Order* dan membuka *form* baru untuk membuat pesanan tambahan
3. *Cancel*: Membatalkan proses pembuatan pesanan

Saat mengklik *create*, sistem akan memproses data dan menghasilkan *batch* dengan kode unik (contoh: FN-OF-FAB/COX-5,900-B00925-251215-0239) yang mengidentifikasi pesanan tersebut secara unik di sistem.

G.3 Proses Approval Factory Order

Setelah *Factory Order* dibuat dengan status *pending*, *GMA* kemudian melakukan proses *approval* untuk memberikan persetujuan resmi kepada *FAB* untuk memulai produksi.

G.3.1 Tampilan Detail Factory Order (Status Pending)

Saat *GMA* membuka detail pesanan *Factory Order* dengan status *pending*, tampilan menunjukkan beberapa *section* penting dengan informasi komprehensif:

– *General Info section*:

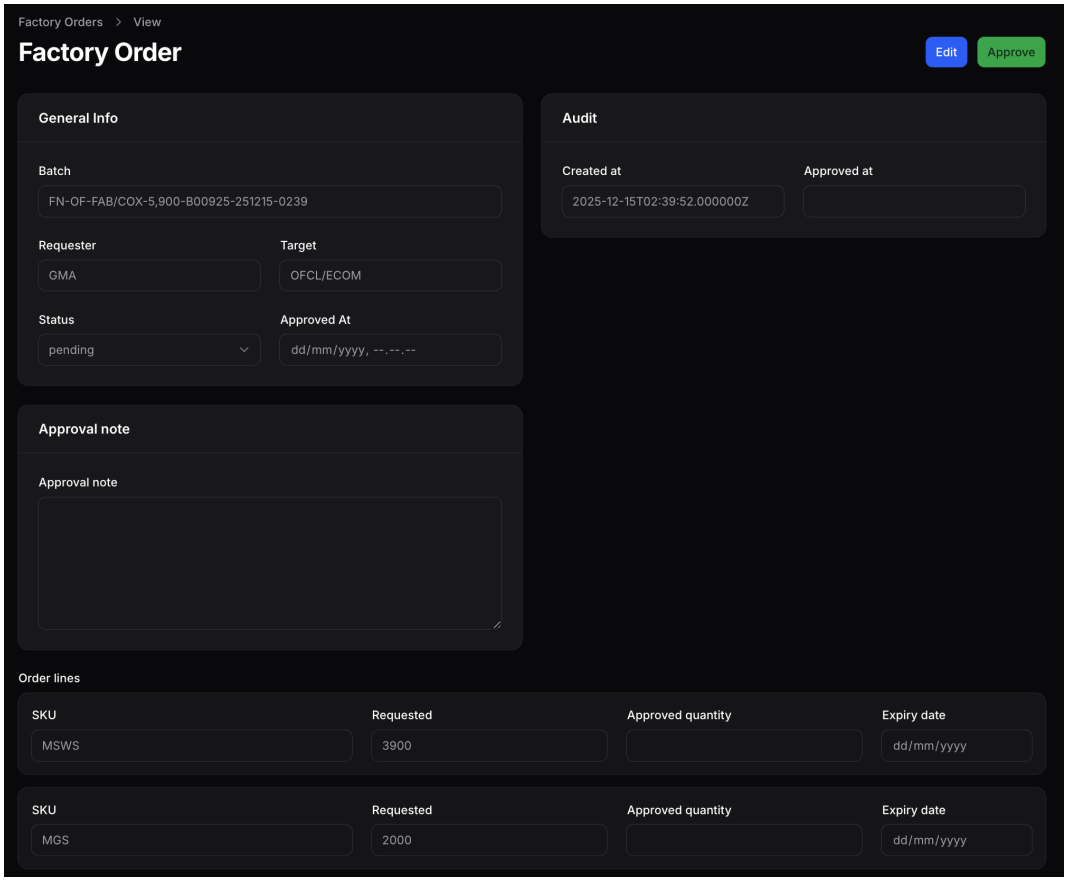
- *Batch*: Kode *batch* unik pesanan (contoh: *FN-OF-FAB/COX-5,900-B00925-251215-0239*)
- *Requester*: Lokasi yang membuat pesanan, dalam hal ini *GMA*
- *Target*: Lokasi tujuan pesanan, dalam hal ini *OFCL/ECOM* (jika pesanan untuk kantor *e-commerce*) atau lokasi lainnya
- *Status*: Menampilkan status saat ini, yaitu *pending* dengan *dropdown* untuk perubahan status (jika diizinkan)
- *Approved At*: *Field* untuk mencatat waktu *approval* (awalnya kosong karena belum di-*approve*)

– *Approval Note section*: *Field* teks besar untuk menambahkan catatan *approval* yang bersifat opsional, biasanya berisi justifikasi atau instruksi khusus terkait pesanan

– *Order Lines section*: Menampilkan detail *item* yang dipesan dengan kolom *SKU*, *requested quantity*, *approved quantity*, dan *expiry date* dalam format *table*

– *Audit section*: Menampilkan waktu pembuatan pesanan (*created at*) dan waktu *approval* (setelah di-*approve*) dengan *timestamp* lengkap

Tampilan *detail Factory Order* dapat dilihat pada Gambar3.30.



Gambar 3.30. *Detail Factory Order* dengan status *pending* menampilkan informasi umum, *order lines*, dan *audit trail*

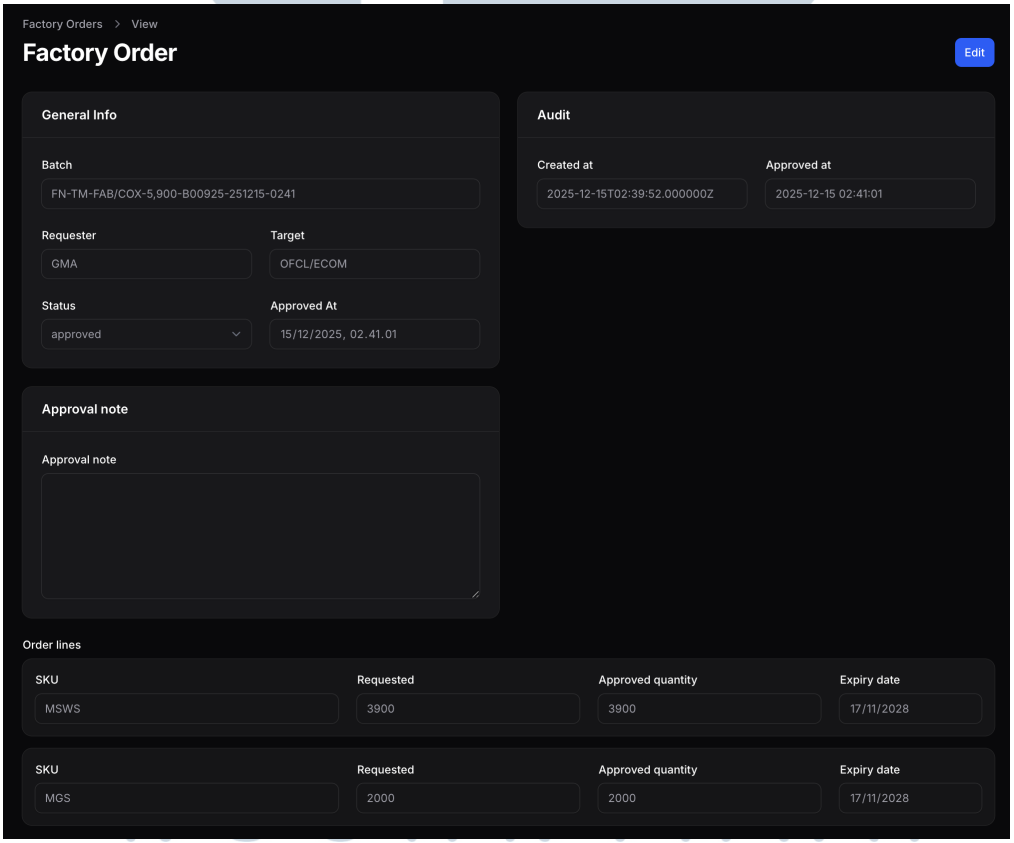
G.3.2 Tombol Approve dan Mekanisme Approval

GMA dapat melakukan *approval* dengan mengklik tombol *approve* (berwarna hijau) yang tersedia di tampilan detail pesanan. Saat tombol *approve* diklik, sistem menampilkan *dialog* konfirmasi dengan pertanyaan apakah yakin melakukan tindakan ini dan beberapa *field* yang harus diisi:

1. *approval note* (*optional*): *GMA* dapat menambahkan catatan *approval* dalam *text area* yang menjelaskan alasan persetujuan atau instruksi khusus untuk *FAB* (contoh: *urgent stock*, *target delivery by* 25 Dec 2025)
2. *confirm lines* (*set approved qty* & *expiry*): *GMA* harus memverifikasi dan mengkonfirmasi setiap *item* dalam pesanan dengan mengisi:
    - *SKU*: Identitas produk (otomatis terisi dari pesanan, misalnya *MSWS* dan *MGS*)

- *Requested*: Jumlah yang diminta (otomatis terisi dari pesanan, contoh: 3900 untuk *MSWS*, 2000 untuk *MGS*)
- *Approved quantity*: Jumlah yang disetujui untuk diproduksi. *GMA* dapat mengubah nilai ini jika diperlukan penyesuaian (contoh: memperketat atau menambah kuantitas berdasarkan kondisi stok terkini)
- *Expiry date (required)*: Tanggal kadaluarsa yang diinginkan dalam format *dd/mm/yyyy*. *Field* ini wajib diisi (ditandai dengan *asterisk* merah) karena akan menjadi ekspirasi *batch* yang dihasilkan dari produksi *FAB*

GMA dapat menambahkan *multiple confirm lines* jika pesanan terdiri dari beberapa item dengan mengklik tombol *add to confirm lines (set approved quantity & expiry)* untuk menambah baris konfirmasi baru. Tampilan tombol *approve* serta mengatur jumlah barang yang disetujui dapat dilihat pada Gambar 3.31.



Gambar 3.31. *Dialog approve* dengan *approval note* dan *confirm lines* untuk verifikasi *item* yang disetujui

G.3.3 Finalisasi Approval

Setelah semua *item* telah diverifikasi dan data *approval* sudah lengkap (*approval note* terisi dan semua *confirm lines* sudah mempunyai nilai *approved quantity* dan *expiry date*), *GMA* mengklik tombol *confirm* (berwarna hijau) untuk menyelesaikan proses *approval*. Saat ini:

- Status pesanan berubah dari *pending* menjadi *approved*
- *Field approved at* diisi dengan *timestamp* saat *approval* (contoh: 15/12/2025, 02.41.01)
- Tombol *approve* berubah menjadi tombol *edit* (berwarna biru), menandakan bahwa pesanan sudah di-*approve* dan tidak dapat diubah lagi
- Sistem mencatat informasi *approval* di *section audit* dengan detail waktu pembuatan (*created at*) dan waktu *approval* (*approved at*)
- Pesanan *Factory Order* siap diteruskan ke *FAB* untuk proses produksi dengan spesifikasi dan *approval* data yang telah tersimpan

G.4 Status dan Alur Factory Order

Factory Order memiliki beberapa status utama yang menunjukkan tahapan pesanan dalam *lifecycle*-nya:

G.4.1 Status dalam Lifecycle Factory Order

1. *Pending*: Pesanan baru telah dibuat oleh *GMA* namun belum mendapat persetujuan. Pada status ini, pesanan hanya bersifat *request* dan belum menjadi komitmen produksi ke *FAB*. *GMA* dapat meng-*edit* atau menghapus pesanan pada tahap ini
2. *Approved*: Pesanan telah disetujui oleh *GMA* dengan verifikasi lengkap dan data *approval* sudah tersimpan. Status ini menandakan bahwa *FAB* dapat memulai proses produksi sesuai spesifikasi yang telah diapprove. Pada status ini, pesanan bersifat *final* dan tidak dapat diubah lagi
3. *Production*: (Jika ada) Menunjukkan bahwa *FAB* sedang melakukan proses produksi untuk *batch* yang dipesan. Status ini mungkin ditambahkan untuk

tracking lebih detail di *future development* untuk *visibilitas* produksi yang lebih baik

4. *Completed*: Produksi selesai dan *batch* telah diterima di *GMA* dengan verifikasi kualitas dan kuantitas. Pada tahap ini, stok baru masuk ke *inventory GMA* dan siap untuk didistribusikan ke lokasi-lokasi lainnya sesuai kebutuhan

G.5 Alur Data dari Factory Order hingga Stock Batch

Setelah *Factory Order* diapprove, alur proses berlanjut hingga stok fisik tiba dan tercatat di sistem sebagai *stock batch*:

G.5.1 Alur Lengkap Factory Order

Berikut adalah alur lengkap dari pembuatan *Factory Order* hingga terbentuknya *stock batch* di *GMA*, seperti yang ditampilkan pada Gambar 3.32:

Gambar 3.32. Alur lengkap *Factory Order* dari pembuatan pesanan hingga terbentuknya *stock batch* di *GMA*

Setiap tahap dalam alur ini tercatat dengan detail untuk keperluan *audit* dan *transparency* dalam sistem manajemen stok *S2G Stock WebApp*. Data dari setiap tahap disimpan dan dapat diakses untuk analisis dan pelaporan.

G.6 Keuntungan Model Centralized Approval oleh GMA

Model di mana *GMA* melakukan kedua peran (*requestor* dan *approver*) memberikan beberapa keuntungan operasional yang signifikan:

G.6.1 Keuntungan Implementasi

– *Single Point of Truth*: Semua data pesanan dan *approval* terpusat di satu akun *GMA*, sehingga tidak ada ambiguitas atau duplikasi informasi dalam sistem. Hal ini memudahkan untuk *maintain data integrity*

– *Response Time* Cepat: *GMA* dapat langsung meng-*approve* atau menolak pesanan tanpa perlu *waiting time* untuk koordinasi dengan pihak ketiga, sehingga *production* dapat dimulai dengan cepat

– *Flexibility* dalam *Adjustment*: *GMA* dapat menyesuaikan kuantitas atau tanggal ekspirasi pada saat *approval* berdasarkan kondisi stok terkini tanpa perlu renegosiasi dengan pihak lain

– *Clear Accountability*: *Responsibility* jelas terletak pada *GMA*, memudahkan identifikasi penyebab masalah jika ada *discrepancy* antara pesanan dan penerimaan dari *FAB*

– *Reduced Administrative Overhead*: Mengurangi beban administratif dan komunikasi dengan eliminasi *approval layer* tambahan dari pihak yang tidak memiliki akses sistem *website*

G.7 Integrasi dengan Batch Penerimaan

Setelah *Factory Order* diapprove dan *FAB* melakukan produksi sesuai spesifikasi yang telah disetujui, hasil produksi dikirimkan ke *GMA* dalam bentuk *batch*. Batch ini akan dicatat sebagai *stock batch* baru di *GMA* dengan informasi terstruktur:

– *Batch Code*: Kode *batch* yang di-*generate* dari *Factory Order reference* (contoh: B00125/251208)

– *Source*: *FAB/CH* (sumber produksi dari pabrik)

– *Target*: *GMA* (lokasi penerimaan di pusat distribusi)

– *Items Received*: Setiap *item* dari *Factory Order* akan menjadi *stock item* dengan kategori dan sub-kategori sesuai tipe pabrik yang dipilih (contoh: kategori *BSK* untuk barang sempurna)

– *Expiry Date*: Tanggal kadaluarsa yang telah ditentukan saat *approval* (contoh: 17/11/2028)

– *Status*: *Received* dan *ready* untuk didistribusikan ke lokasi lain atau dikategorisasi berdasarkan kondisi fisik sesuai *quality control*

Dengan mekanisme integrasi ini, *Factory Order* dan *stock batch* terintegrasi *seamlessly* dalam sistem, menciptakan *audit trail* yang lengkap dari proses permintaan produksi hingga penerimaan stok di gudang utama. *Traceability* yang sempurna ini memastikan bahwa setiap *batch* dapat dilacak origin-nya dan setiap keputusan *approval* dapat diaudit dengan jelas.

H Stock Distribution (Distribusi Stok)

Stock Distribution adalah proses pendistribusian stok dari satu *batch* ke berbagai kategori dan sub-kategori sesuai kebutuhan dan tujuan distribusi. Sistem ini memungkinkan lokasi manapun (*GMA*, *GDB*, *GOF*, *DBW*, atau lokasi lainnya) untuk membagi *batch* besar menjadi segmen-segmen yang lebih kecil dengan kategori berbeda (*BSK*, *E*, *BR*) dan sub-kategori berbeda (*FN*, *BT*, *BX*) untuk dialokasikan ke berbagai keperluan yang spesifik.

H.1 Konsep Stock Distribution dan Penerapannya di Setiap Lokasi

Stock Distribution adalah mekanisme alokasi stok yang memungkinkan *batch* induk untuk dipisahkan menjadi *multiple stock items* dengan kategori dan sub-kategori yang berbeda. Proses ini penting untuk memastikan bahwa setiap *item* stok dapat diidentifikasi dengan detail sesuai kondisi dan tujuan penggunaannya. Sistem ini dirancang agar universal dan dapat diterapkan di semua akun lokasi, tidak hanya di *GMA* saja.

H.1.1 Penerapan Stock Distribution di GDB, GOF, dan DBW

Stock Distribution dapat digunakan oleh semua lokasi dalam sistem untuk keperluan *masukin stok* (*receiving* dan *categorizing*) dari *batch* yang diterima:

- *GDB* (Gudang Distribusi *Backup*): Melakukan *stock distribution* ketika menerima *batch* dari *GMA*, untuk segmentasi stok regional sesuai kebutuhan distributor lokal
- *GOF* (Gudang *Office*/Kantor): Melakukan *stock distribution* untuk stok kantor dan *retail*, serta alokasi ke *e-commerce fulfillment center*
- *DBW* (Distributor Jawa) dan Distributor Regional Lainnya: Melakukan *stock distribution* untuk stok penjualan *retail* ke *end consumer*, dengan segmentasi berdasarkan kondisi dan format kemasan

Dengan demikian, proses *Stock Distribution* yang dijelaskan di bawah berlaku sama untuk semua lokasi tersebut. Setiap lokasi menggunakan sistem yang sama untuk menerima, mengkategorisasi, dan mendistribusikan stok sesuai kebutuhan lokal mereka.

H.1.2 Alasan Pentingnya Stock Distribution

Beberapa alasan strategis mengapa *Stock Distribution* diperlukan dalam sistem:

- Kategorisasi Berdasarkan Kondisi: *Batch* yang diterima perlu dikategorisasi menjadi *BSK* (Barang Sempurna), *E* (*Emergency*), atau *BR* (Barang Rusak) berdasarkan kondisi fisik untuk *quality control*
- Segmentasi Sub-Kategori: Setiap kategori dapat dibagi lagi menjadi sub-kategori (*FN*, *BT*, *BX*) sesuai format kemasan dan tujuan distribusi
- *Tracking Granular*: Dengan distribusi detail, setiap unit stok dapat dilacak dengan akurat hingga level *stock item* individual
- Alokasi Strategis: Memungkinkan alokasi yang lebih presisi sesuai kebutuhan lokal dengan kategorisasi yang tepat
- *Audit Trail* Lengkap: Setiap distribusi tercatat dengan detail untuk keperluan *audit* dan *accountability* di setiap lokasi

H.2 Proses Masukin Stok Melalui Stock Distribution

Proses *Stock Distribution* adalah cara standar untuk memasukkan stok baru ke dalam sistem setiap kali lokasi menerima *batch* dari lokasi induk atau *supplier*. Proses ini berlaku sama untuk *GDB*, *GOF*, *DBW*, dan lokasi lainnya.

H.2.1 Langkah 1: Akses Modul Create Stock Distribution

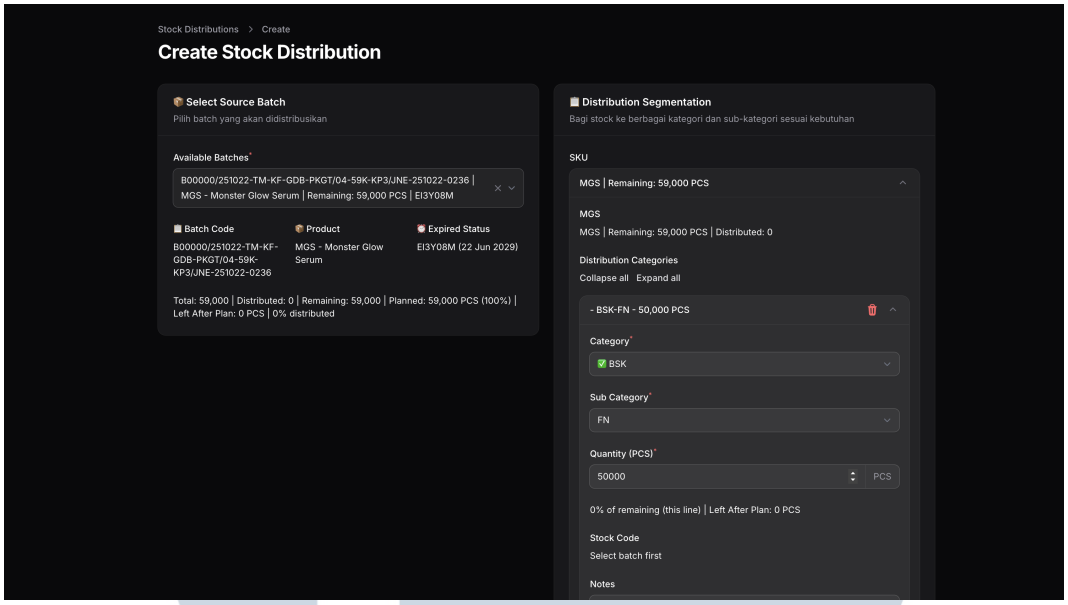
Lokasi (*GDB*, *GOF*, atau *DBW*) membuka modul *create Stock Distribution* di sistem untuk memulai proses masukin stok baru yang telah diterima. Pada tampilan *form*, terdapat dua *section* utama untuk proses penerimaan dan kategorisasi:

1. *Select Source Batch*: Lokasi memilih *batch* yang akan didistribusikan. *Batch* yang tersedia adalah *batch* yang baru diterima atau sudah ada di lokasi tersebut. *Batch* yang tersedia menampilkan informasi lengkap:

- *Batch Code*: Kode *batch* unik (contoh: B000125/251022-TM-KF-GDB-PKGT/04-59K-KP3J/UNE-251022-0236)
- *Product*: Nama produk dalam *batch* (contoh: *MGS - Monster Glow Serum*)
- *Expired Status*: Status kadaluarsa *batch* (contoh: *EI3Y08M* (22 Jun 2029))
- *Total*: Jumlah total stok dalam *batch* (contoh: 59,000 *PCS*)
- *Distributed*: Jumlah yang sudah didistribusikan sebelumnya
- *Remaining*: Jumlah yang masih tersisa untuk didistribusikan (contoh: 59,000 *PCS* untuk *batch* baru yang belum ada kategori)
- *Planned*: Jumlah yang sudah direncanakan untuk didistribusikan

2. *Distribution Segmentation*: Setelah memilih *batch* sumber, lokasi melakukan segmentasi dengan menentukan kategori dan sub-kategori untuk stok yang diterima:
   - *SKU*: Produk yang akan didistribusikan (otomatis terisi dari *batch* yang dipilih)
   - *Distribution Categories*: Kategori distribusi yang tersedia dengan opsi *collapse all* atau *expand all* untuk melihat detail
   - *Category*: Klasifikasi stok berdasarkan kondisi (*BSK*, *E*, *BR*) dengan *dropdown selection*
   - *Sub Category*: Sub-klasifikasi stok (*FN*, *BT*, *BX*) dengan *dropdown selection* sesuai format kemasan
   - *Quantity* (*PCS*): Jumlah unit untuk kategori ini (dengan informasi sisa yang masih bisa dialokasikan)
   - *Stock Code*: Kode *stock* yang akan di-*generate* untuk *item* kategori ini (otomatis atau *manual selection*)
   - *Notes*: *Field* opsional untuk menambahkan catatan khusus distribusi kategori ini

Tampilan modul *create stock distribution* dapat dilihat pada Gambar 3.33.



Gambar 3.33. *Form create Stock Distribution* untuk penerimaan dan kategorisasi stok

H.2.2 Langkah 2: Menambahkan Multiple Distribution Categories

Lokasi dapat menambahkan *multiple* kategori dalam satu *distribution* untuk mengkategorisasi *batch* sesuai kondisi fisik stok yang diterima. Dengan mengklik tombol *add another category*, lokasi dapat membagi *batch* menjadi beberapa kategori. Contoh skenario distribusi stok yang diterima:

- Kategori 1 (*BSK-FN*): *Category BSK* (Barang Sempurna), *Sub Category FN* (*Final/Normal*), *Quantity* 50,000 *PCS* untuk stok dalam kondisi terbaik
- Kategori 2 (*E-FN*): *Category E* (*Emergency*), *Sub Category FN* (*Final/Normal*), *Quantity* 9,000 *PCS* untuk stok yang masih bisa digunakan tapi ada *minor issue*

Sistem akan memvalidasi bahwa total *quantity* yang didistribusikan sesuai dengan *remaining quantity* dari *batch* sumber. Setiap kategori dapat memiliki *notes* terpisah untuk *tracking* lebih detail kondisi stok per kategori.
Catatan penting: Proses kategorisasi ini adalah bagian dari masukin stok karena lokasi sedang mengkategorisasi *batch* yang baru diterima menjadi *stock items* yang lebih spesifik dan terkelompok berdasarkan kondisi sehingga siap digunakan untuk distribusi lebih lanjut atau penjualan.

H.2.3 Langkah 3: Overall Distribution Notes dan Finalisasi Masukin Stok

Setelah semua kategori ditambahkan, lokasi dapat menambahkan *overall distribution notes* sebagai catatan umum untuk seluruh proses masukin stok ini (contoh: catatan kondisi stok saat penerimaan, verifikasi jumlah, dll). Kemudian lokasi memilih aksi akhir:

1. *Create*: Menyelesaikan proses masukin stok dan menyimpan kategorisasi ke sistem (stok menjadi *ready* untuk digunakan)
2. *Create & create another*: Menyelesaikan masukin stok saat ini dan membuka *form* baru untuk *batch* tambahan
3. *Cancel*: Membatalkan proses masukin stok

Saat mengklik *create*, sistem akan memproses data kategorisasi dan menciptakan *multiple stock items* (*stock distribution items*) dengan kategori dan sub-kategori sesuai yang telah ditentukan, sehingga stok formal masuk ke sistem dan *ready* untuk:

- Untuk *GDB*: Siap untuk didistribusikan ke *warehouse order* menuju *DBW* atau lokasi regional lainnya
- Untuk *GOF*: Siap untuk penjualan *retail* atau alokasi ke *e-commerce fulfillment center*
- Untuk *DBW*: Siap untuk penjualan ke *end consumer* atau alokasi ke *outlet retail partner*

H.3 View Stock Distribution (Hasil Masukin Stok)

Setelah *Stock Distribution* dibuat (stok berhasil dimasukkan), lokasi dapat melihat detail distribusi yang menampilkan informasi komprehensif tentang stok yang telah dikategorisasi:

H.3.1 Detail View Stock Distribution

Tampilan detail menampilkan beberapa *section* penting yang memperlihatkan stok yang telah masuk ke sistem:

– *Stock Distribution Details* (Informasi Stok yang Masuk):

  - *SKU Code*: Kode produk (contoh: *MGS*)
  - *Stock Code*: Kode *stock* hasil kategorisasi (contoh: *BSKFN-MGS-EI3Y08M*) ini adalah identitas *stock* yang baru masuk
  - *Category*: Kategori stok yang diinputkan (*BSK*, *E*, *BR*)
  - *Current Stock*: Jumlah stok yang telah masuk ke kategori ini (contoh: 50,000 *PCS*)
  - *Available Stock*: Jumlah stok yang tersedia untuk digunakan (contoh: 50,000 *PCS* berarti belum ada yang dipindahkan)
  - *Expiry Status*: Status kadaluarsa dengan *visual indicator* (*red dot* untuk *expired*)

– *Batch & Product Information* (Informasi Batch Sumber):

  - *Source Batch*: Kode *batch* sumber yang dimasukkan (contoh: *B00000/251022-TM-KF-GDB-PKGT/04-59K-KP3J/UNE-251022-0236*)
  - *Product Type*: Tipe produk dalam distribusi (contoh: *FN Final + Box*)
  - *Factory Code*: Kode lokasi sumber (contoh: *GDB* jika ini masukin stok di *GDB* dari *GMA*)
  - *Product Name*: Nama produk (contoh: *Monster Glow Serum*)

– *Date & Time Information* (*Timeline* Penerimaan):

  - *Batch Expired Date*: Tanggal kadaluarsa *batch* (contoh: 22 Jun 2029)
  - *Item Expired Date*: Tanggal kadaluarsa *item* dari distribusi (contoh: 22 Jun 2029 sama dengan *batch*)
  - *Time Remaining*: Sisa waktu sebelum kadaluarsa dengan *visual indicator* (contoh: *3 years from now*)

– *Additional Information* (Info Penerimaan):

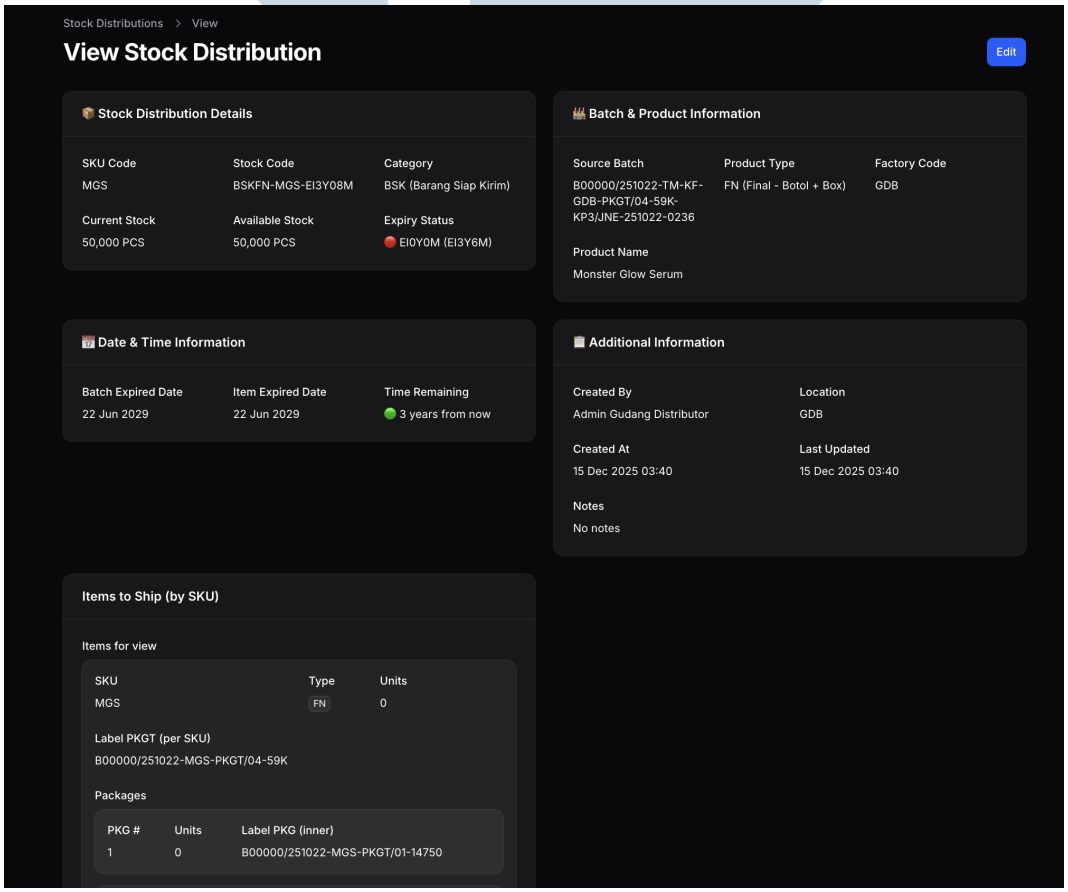
  - *Created By*: *User* yang melakukan proses masukin stok (contoh: Admin Gudang Distributor *GDB*)
  - *Location*: Lokasi tempat stok dimasukkan (contoh: *GDB*, *GOF*, atau *DBW*)

- *Created At*: *Timestamp* kapan stok dimasukkan ke sistem (contoh: 15 Dec 2025 03:40)
- *Last Updated*: *Timestamp update* terakhir pada stok ini
- *Notes*: Catatan distribusi yang diinputkan saat masukin stok

– *Items to Ship* (by *SKU*) (Detail Stok yang Masuk):

- *Items for view*: Menampilkan detail *SKU* dan jumlah unit yang sudah masuk
- *Label PKGT*: Kode label *packaging* untuk *tracking* (jika akan dikirim ke lokasi lain)
- *Packages*: Detail *packaging* dengan *package number*, *units per package*, dan *label inner* untuk *tracking shipment*

Tampilan *detail view stock distribution* dapat dilihat pada Gambar3.34.



Gambar 3.34. *Detail view Stock Distribution* menampilkan informasi stok yang masuk dengan kategori dan kondisi

H.4 Stock Distribution List (Monitor Stok yang Sudah Masuk)

Lokasi dapat melihat daftar semua *Stock Distribution* yang telah dibuat (semua stok yang sudah dimasukkan) dengan berbagai fitur *filtering* dan *sorting* untuk memantau stok per lokasi:

H.4.1 Tampilan List Stock Distribution

Halaman *list* menampilkan informasi ringkas tentang semua stok yang telah dimasukkan ke dalam sistem lokasi tersebut dengan fitur-fitur monitoring:

– *Summary Information* (Ringkasan Stok Lokasi):

- *Total Stock*: Total stok dalam semua distribusi di lokasi ini (contoh: 59,000) ini adalah total stok yang sudah masuk
- *Available*: Stok yang tersedia untuk digunakan (contoh: 59,000 berarti semua stok masih tersedia belum dipakai)
- *Expired*: Jumlah stok yang sudah kadaluarsa (contoh: 0 berarti tidak ada stok *expired*)
- *Critical*: Jumlah stok dalam status kritis mau kadaluarsa atau perlu *alert* (contoh: 2)

– *Action Buttons*:

- ***+ New Distribution***: Tombol biru untuk masukin stok baru (*batch* berikutnya yang diterima)
- ***Expiry Dashboard***: Tombol *orange* untuk melihat *dashboard* stok yang hampir kadaluarsa
- ***SKU Breakdown***: Tombol biru untuk melihat *breakdown* stok per *SKU* di lokasi ini

– *Filtering* & *Sorting* (Untuk Monitor Stok):

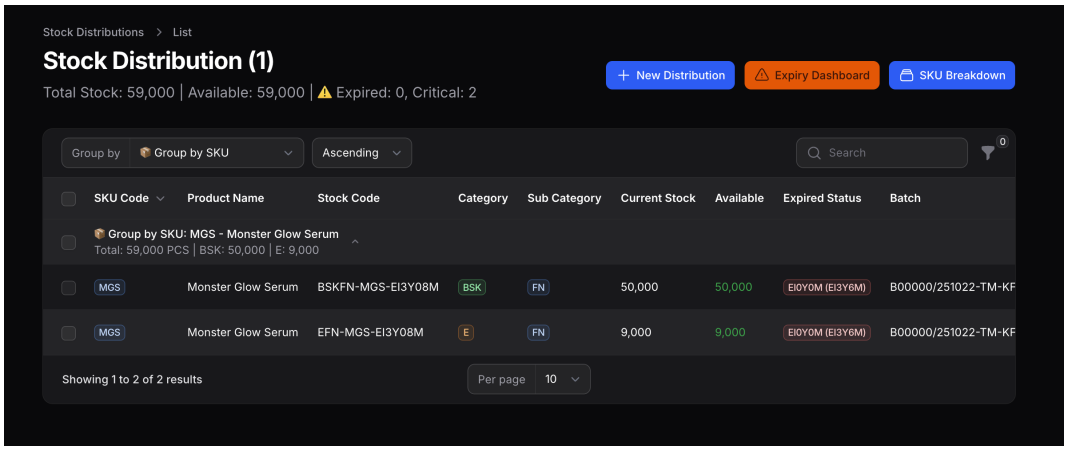
- *Group by*: *Dropdown* untuk *grouping* data (contoh: *Group by SKU* untuk melihat stok per produk)
- *Ascending*/*Descending*: Pilihan untuk *sort order* berdasarkan kolom tertentu

- *Search*: *Field* untuk mencari distribusi atau stok tertentu

– *Table Columns* (Informasi Stok Terpusat):

- *SKU Code*: Kode produk
- *Product Name*: Nama produk yang sudah masuk
- *Stock Code*: Kode *stock* dari kategori yang dimasukkan
- *Category*: Kategori stok yang diinputkan (dengan *badge color-coded*: *green* untuk *BSK*, *yellow* untuk *E*, *red* untuk *BR*)
- *Sub Category*: Sub-kategori stok (*FN*, *BT*, *BX*)
- *Current Stock*: Jumlah stok saat ini di kategori ini
- *Available*: Jumlah stok tersedia (dalam warna hijau jika sama dengan *current stock*, *orange* jika ada yang sudah dipakai)
- *Expired Status*: Status kadaluarsa dengan *visual indicator* dan kode (contoh: *EI0Y0M* berarti *Expired In 0 Years 0 Months* sudah *expired*)
- *Batch*: Kode *batch* sumber stok ini

Tampilan *list stock distribution* dapat dilihat pada Gambar 3.35.



Gambar 3.35. *List Stock Distribution* monitor stok yang sudah dimasukkan dengan *summary* informasi dan detail kategori

I Stock Batches (Batch Stock)

Stock Batches adalah kumpulan stok dalam *batch* yang merupakan hasil dari penerimaan dari lokasi induk (*GMA*, *GDB*, *GOF*) atau *supplier* lainnya. Setiap

batch memiliki identitas unik dan melacak keseluruhan *lifecycle* stok dari penerimaan hingga distribusi dan penggunaan. Setiap lokasi (*GDB*, *GOF*, *DBW*) memiliki *batch*-nya sendiri yang telah diterima.

I.1 Konsep Stock Batches

Stock Batches adalah unit dasar dalam sistem manajemen stok yang merepresentasikan stok dalam jumlah besar dengan identitas *batch* yang sama. Setiap *batch* memiliki *batch* code unik, tanggal kadaluarsa, dan status yang menunjukkan tahap *lifecycle* stok tersebut. *Stock Batches* adalah rekam jejak stok sebelum dilakukan *Stock Distribution* (kategorisasi).

I.1.1 Alasan Pentingnya Stock Batches

Beberapa alasan mengapa *Stock Batches* menjadi fondasi sistem manajemen stok:

- *Identification & Traceability*: Setiap *batch* dapat dilacak hingga sumbernya (*FAB*, *GMA*, atau lokasi lainnya) dengan *batch code* yang unik
- *Expiry Management*: Sistem dapat melacak tanggal kadaluarsa per *batch* untuk *quality control* dan *alert management*
- *Quantity Tracking*: Memudahkan *tracking* jumlah stok total, *remaining* (belum dikategorisasi), dan *distributed* (sudah dikategorisasi) per *batch*
- *Distribution Planning*: *Batch* besar dapat direncanakan untuk didistribusikan (dikategorisasi) ke berbagai kategori
- *Audit Trail*: Setiap *batch* memiliki *history* pembuatan, distribusi, dan status perubahan yang ter-*record* lengkap

I.2 List Stock Batches (Monitor Batch Stok)

Setiap lokasi dapat melihat halaman list *Stock Batches* yang menampilkan semua *batch* yang telah diterima atau tersedia di lokasi tersebut dengan informasi detail dan opsi aksi untuk setiap *batch*:

I.2.1 Tampilan List Stock Batches

Halaman ini menampilkan tabel dengan informasi *batch* yang diterima dan aksi yang dapat dilakukan per *batch*:

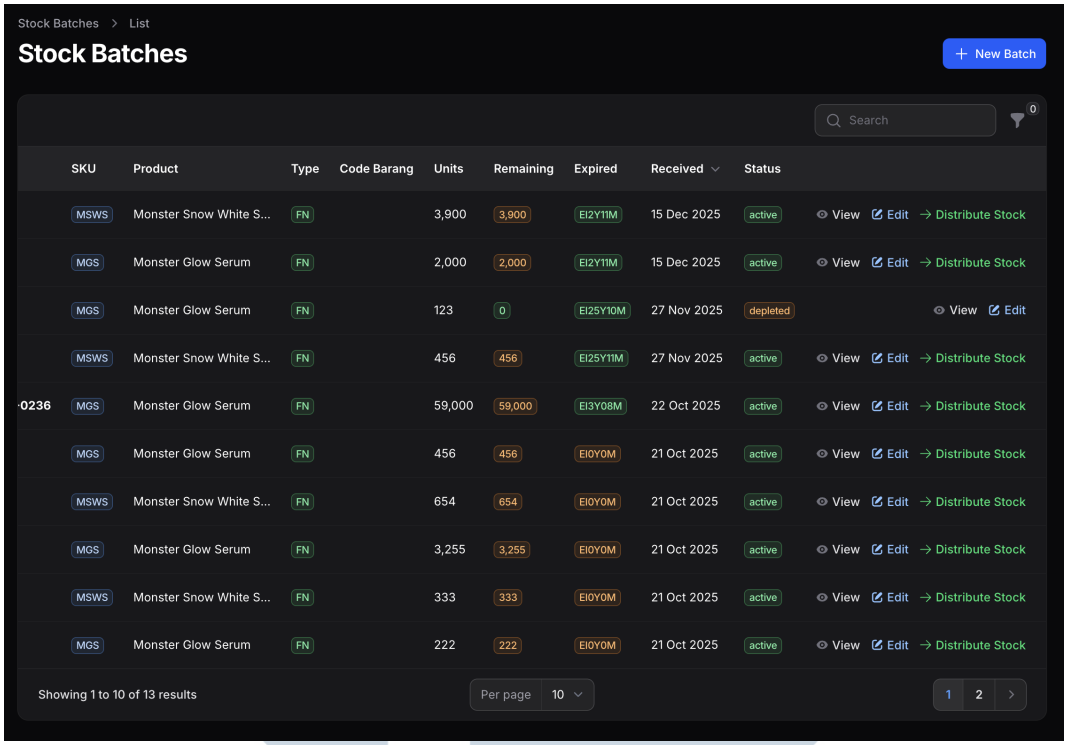
– *Table Columns* (Informasi *Batch*):

- *SKU*: Kode produk (contoh: *MSWS* untuk *Monster Snow White Serum*, *MGS* untuk *Monster Glow Serum*)
- *Product*: Nama lengkap produk (contoh: *Monster Snow White S...*, *Monster Glow Serum*)
- *Type*: Tipe *batch* (*FN* untuk *Final/Normal*, *BT* untuk *Batch/Bulk*, *BX* untuk *Box*)
- *Code Barang*: Kode identifikasi *batch* dan *expiry status* (contoh: *EI2Y1TM = Expired In 2 Years 1 Month*)
- *Units*: Total jumlah unit dalam *batch* saat diterima
- *Remaining*: Jumlah unit yang masih belum dikategorisasi melalui *Stock Distribution*
- *Expired*: Kode status kadaluarsa *batch* secara keseluruhan
- *Received*: Tanggal penerimaan *batch* di lokasi ini
- *Status*: Status *batch* (*active* = masih ada stok, *depleted* = sudah habis, *completed* = sudah dikategorisasi semua)

– *Action Buttons* per *Batch* (Opsi Aksi):

- *View*: Tombol untuk melihat detail *batch* dan *history* distribusinya
- *Edit*: Tombol untuk mengedit informasi *batch* (jika diperlukan)
- *Distribute Stock*: Tombol hijau untuk membuat *stock distribution* dari *batch* ini (untuk masukin stok yang belum dikategorisasi)

Tampilan *stock batches* dapat dilihat pada Gambar3.48.



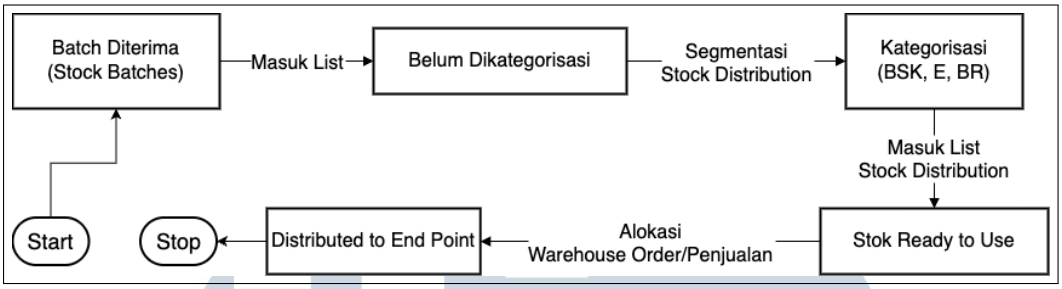
Gambar 3.36. *List Stock Batches* monitor *batch* yang diterima di lokasi dengan informasi stok dan status kategorisasi

I.3 Alur Lengkap Masukin Stok: Dari Penerimaan Batch hingga Kategorisasi

Berikut adalah alur lengkap proses masukin stok di setiap lokasi (*GDB*, *GOF*, *DBW*), dari penerimaan *batch* hingga kategorisasi melalui *Stock Distribution*:

I.3.1 End-to-End Flow Masukin Stok

Berikut adalah alur lengkap dari penerimaan *batch* hingga distribusi ke *end point*, seperti yang ditampilkan pada Gambar 3.37:



Gambar 3.37. Alur *end-to-end* masukin stok dari penerimaan *batch* hingga distribusi ke end point

Proses ini sama untuk semua lokasi dengan tahapan yang konsisten:

- *GDB*: Terima *batch* dari *GMA* → Kategorisasi via *Stock Distribution* → Alokasikan ke *DBW* atau lokasi regional via *warehouse order*
- *GOF*: Terima *batch* dari *GMA* → Kategorisasi via *Stock Distribution* → Alokasikan ke stok kantor atau *e-commerce fulfillment*
- *DBW*: Terima *batch* dari *GDB* → Kategorisasi via *Stock Distribution* → Alokasikan untuk penjualan *retail* ke *end consumer*

I.3.2 Kontrol Kualitas dalam Proses Masukin Stok

Saat melakukan *Stock Distribution* (kategorisasi), setiap lokasi melakukan kontrol kualitas dengan memverifikasi kondisi stok dan menentukan kategori yang tepat:

- *BSK* (Barang Sempurna): Stok dalam kondisi terbaik, siap untuk penjualan atau distribusi langsung
- *E* (*Emergency*): Stok dengan *minor issue* tapi masih bisa digunakan, untuk keperluan mendesak saja
- *BR* (Barang Rusak): Stok yang tidak memenuhi standar, perlu evaluasi lebih lanjut atau *disposal*

Kategorisasi ini dilakukan saat *Stock Distribution*, memastikan hanya stok berkualitas yang dialokasikan untuk distribusi lebih lanjut.

I.4 Integrasi dengan Warehouse Order untuk Distribusi Lanjutan

Setelah *Stock Distribution* dibuat (stok berhasil dikategorisasi dan masuk ke sistem), stok yang telah disegmentasi dapat digunakan untuk:

- Untuk *GDB*: Membuat *warehouse order* ke *DBW* atau distributor regional lainnya berdasarkan stok *BSK* dan *E* yang sudah dikategorisasi
- Untuk *GOF*: Mengalokasikan stok ke *e-commerce fulfillment center* atau stok *retail* berdasarkan kategori yang dibutuhkan
- Untuk *DBW*: Menggunakan stok untuk penjualan *retail* langsung ke *end consumer* atau alokasi ke *outlet partner*

Dengan mekanisme *Stock Distribution* dan *Stock Batches* yang terstruktur dan universal untuk semua lokasi, sistem memastikan bahwa:

- Setiap *batch* stok dapat dilacak dari penerimaan hingga distribusi akhir
- Setiap *item* stok dapat diidentifikasi dengan detail (kategori, sub-kategori, *expiry*)
- Setiap lokasi dapat melakukan masukin stok dengan proses yang sama dan terstandar
- *Quality control* dapat dilakukan secara konsisten di semua lokasi
- *Audit trail* lengkap tersedia untuk *compliance* dan *accountability*

J Warehouse Order (Pesanan Gudang)

Warehouse Order adalah transaksi distribusi stok antar lokasi dalam jaringan distribusi yang memungkinkan lokasi untuk memesan stok dari lokasi induk atau lokasi lain sesuai kebutuhan inventaris lokal. Sistem ini dirancang dengan mekanisme *approval* terpusat untuk menjaga kontrol stok dan memastikan alur distribusi berjalan secara terorganisir dan transparan di setiap tingkatan hierarki distribusi.

J.1 Peran dan Fungsi Warehouse Order

Warehouse Order memiliki peran krusial dalam sistem distribusi karena menghubungkan berbagai lokasi dengan mekanisme permintaan dan persetujuan yang jelas. *Warehouse Order* memungkinkan setiap lokasi untuk mengajukan permintaan stok dan memberikan visibilitas penuh tentang alur distribusi dari pusat hingga ke lokasi akhir.

J.1.1 Alur Universal Warehouse Order

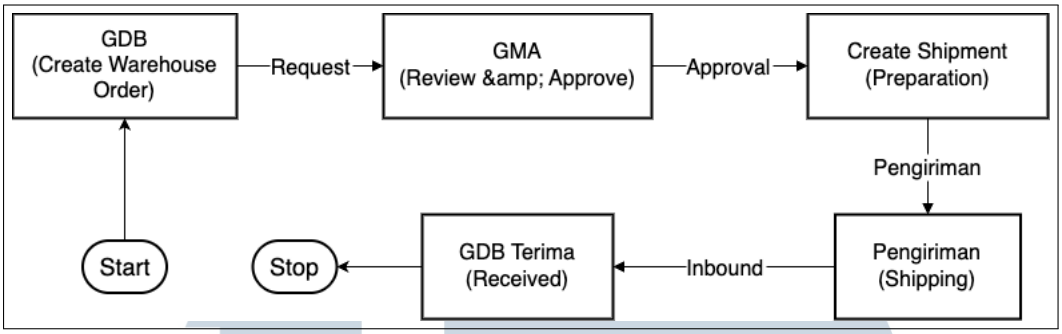
Alur *Warehouse Order* berlaku secara universal untuk semua jalur distribusi dalam sistem:

- *GDB* $\rightarrow$ *GMA*: *Hub* distribusi regional memesan stok dari pusat distribusi utama
- *DBW/JKT* $\rightarrow$ *GDB*: Distributor regional memesan stok dari *hub* distribusi regional
- *GOF* $\rightarrow$ *GMA*: Kantor dan *retail* memesan stok dari pusat distribusi utama
- *ECOM/TXT*, *ECOM/SH* $\rightarrow$ *OFCL/ECOM*: *Fulfillment center e-commerce* memesan stok dari pusat kontrol *e-commerce*

Setiap alur memiliki proses yang sama: *request* stok, *approval* oleh lokasi induk, pembuatan *shipment*, dan penerimaan di lokasi tujuan. Dengan mekanisme universal ini, sistem memastikan konsistensi dalam pengelolaan pesanan di semua tingkatan hierarki.

J.1.2 Contoh Alur GDB ke GMA

Berikut adalah contoh alur *Warehouse Order* dari *GDB* (*Hub* Distribusi Regional) ke *GMA* (Pusat Distribusi Utama) yang menunjukkan proses detail dari permintaan hingga penerimaan, seperti yang ditampilkan pada Gambar 3.38:



Gambar 3.38. Alur *Warehouse Order* dari *GDB* ke *GMA* menunjukkan proses dari pembuatan pesanan hingga penerimaan barang

Alur ini mendemonstrasikan bagaimana *GDB* sebagai *hub* regional melakukan *request* stok ke *GMA* sebagai pusat distribusi utama, *GMA* melakukan *review* dan *approval* berdasarkan ketersediaan stok, kemudian *GMA* membuat *shipment* untuk pengiriman ke *GDB*, dan akhirnya *GDB* menerima barang untuk kemudian didistribusikan ke lokasi-lokasi regional lainnya atau distributor akhir.

J.2 Proses Pembuatan Warehouse Order

Proses pembuatan *Warehouse Order* dimulai dari lokasi yang membutuhkan stok (misalnya *GDB*) yang mengajukan permintaan stok kepada lokasi induknya (misalnya *GMA*). Proses ini berlaku sama untuk semua jalur distribusi di sistem.

J.2.1 Langkah 1: Akses Modul Create Warehouse Order

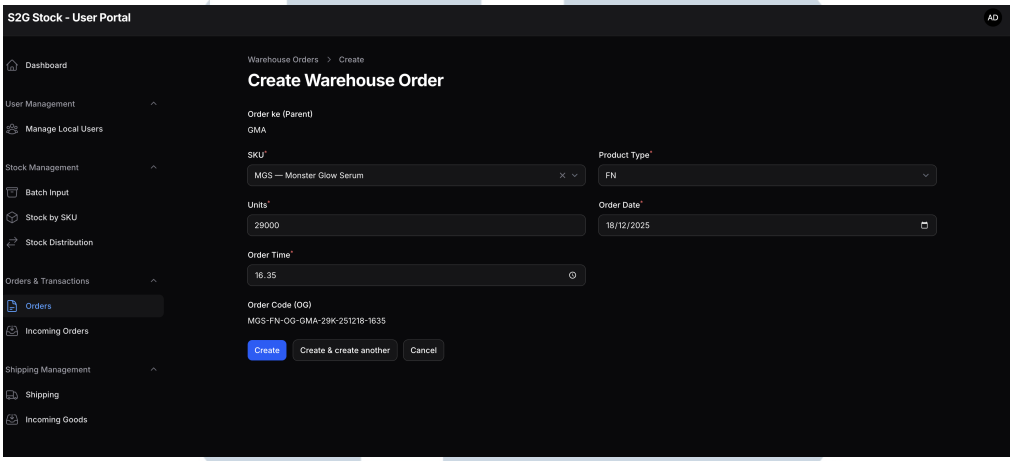
Lokasi yang membutuhkan stok membuka modul *create Warehouse Order* di sistem. Pada tampilan form, lokasi diminta untuk mengisi informasi permintaan stok dengan detail yang komprehensif. Form ini menampilkan beberapa section utama:

– *Order Information Section*:

- *Order from* (*Parent Location*): Lokasi induk yang akan mengirimkan stok (otomatis terisi berdasarkan hierarki lokasi, contoh: *GMA* untuk *GDB*)
- *SKU*: Produk yang diminta dengan kode unik (contoh: *MGS* untuk *Monster Glow Serum*, *MSWS* untuk *Monster Snow White Serum*)
- *Product Type*: Tipe produk yang diminta (*FN*, *BT*, *BX*)

- *Units*: Jumlah unit yang diminta (contoh: 29000 untuk pesanan tertentu)
- *Order Date & Order Time*: Tanggal dan waktu pemesanan dicatat otomatis oleh sistem

Tampilan *form create Warehouse Order* dapat dilihat pada Gambar 3.39.



Gambar 3.39. *Form create Warehouse Order* untuk mengajukan permintaan stok ke lokasi induk

J.2.2 Langkah 2: Konfirmasi dan Pembuatan Pesanan

Setelah mengisi semua informasi permintaan stok, lokasi memiliki tiga opsi aksi:

1. *Create*: Membuat *Warehouse Order* dan menyimpannya ke sistem dengan status *pending* untuk approval lokasi induk (tombol berwarna biru)
2. *Create & create another*: Membuat *Warehouse Order* dan membuka *form* baru untuk pesanan tambahan
3. *Cancel*: Membatalkan proses pembuatan pesanan

Saat mengklik *create*, sistem akan memproses data dan menghasilkan kode pesanan unik (*Order Code* atau *OG Code*) yang mengidentifikasi pesanan tersebut secara unik di sistem (contoh: *MGS-FN-OG-GMA-29K-251218-1635*).

J.3 Proses Approval Warehouse Order

Setelah *Warehouse Order* dibuat dengan status *pending*, lokasi induk melakukan proses *approval* untuk memberikan persetujuan dan memastikan stok tersedia di lokasi induk.

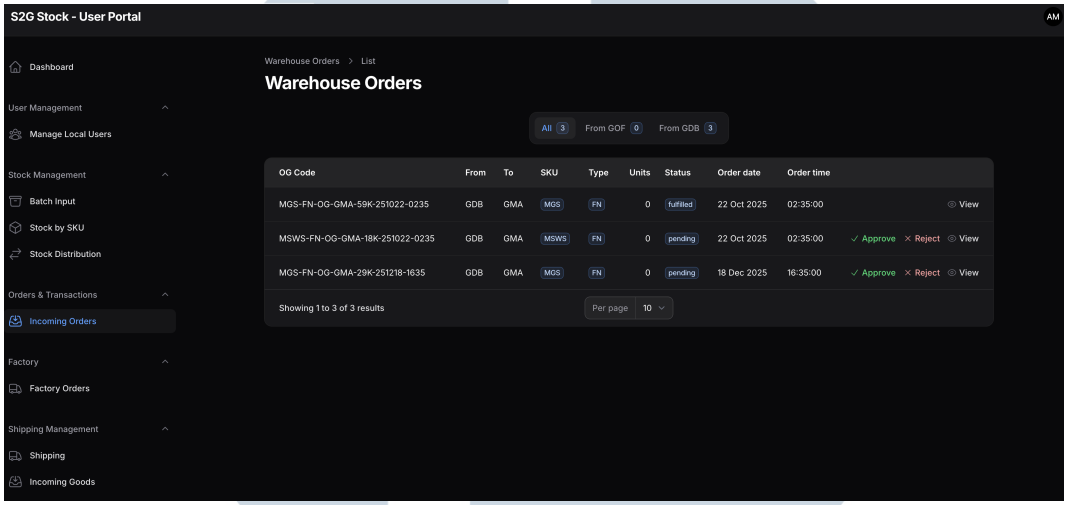
J.3.1 Tampilan List Warehouse Order (Status Pending dan Action Buttons)

Lokasi induk dapat melihat daftar semua *Warehouse Order* yang masuk dengan status berbeda-beda. Tampilan ini menampilkan informasi ringkas tentang semua pesanan dengan berbagai fitur *filtering*:

- *Tab Filtering*: Sistem menyediakan *tab* untuk memfilter pesanan berdasarkan status atau asal pesanan:
  - *All*: Menampilkan semua pesanan (contoh: 3 pesanan total)
  - *From GOF*: Menampilkan pesanan dari *GOF* saja (contoh: 0 pesanan)
  - *From GDB*: Menampilkan pesanan dari *GDB* saja (contoh: 3 pesanan)
- *Table Columns* (Informasi Pesanan):
  - *OG Code*: Kode pesanan unik (contoh: *MGS-FN-OG-GMA-59K-251022-0235*)
  - *From*: Lokasi pengirim pesanan (contoh: *GDB*)
  - *To*: Lokasi penerima pesanan (contoh: *GMA*)
  - *SKU*: Kode produk yang dipesan (contoh: *MGS*, *MSWS*)
  - *Type*: Tipe produk (*FN*, *BT*, *BX*)
  - *Units*: Jumlah unit yang dipesan (contoh: 0 berarti masih dalam *processing*)
  - *Status*: Status pesanan (*fulfilled*, *pending*, *approved*) dengan *visual indicator*
  - *Order date*: Tanggal pesanan dibuat (contoh: 22 Oct 2025)
  - *Order time*: Waktu pesanan dibuat (contoh: 02:35:00)
- *Action Buttons* (Opsi Aksi per Pesanan):
  - *View*: Tombol untuk melihat detail pesanan
  - *Approve*: Tombol hijau untuk menyetujui pesanan (muncul saat status *pending*)
  - *Reject*: Tombol merah untuk menolak pesanan (muncul saat status *pending*)

- *Create Shipment*: Tombol untuk membuat *shipment* setelah pesanan di-*approve* (muncul saat status *approved*)

Tampilan *list Warehouse Order* dapat dilihat pada Gambar 3.40.



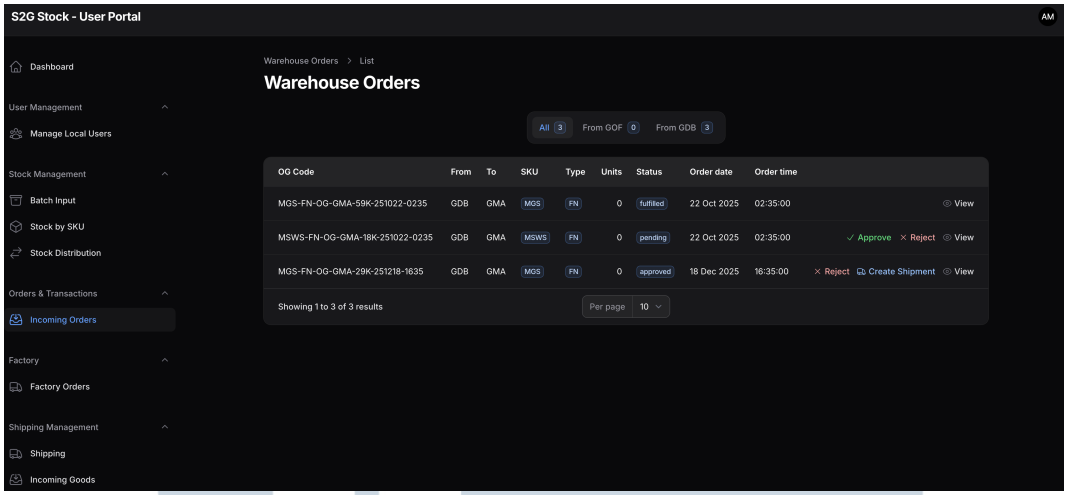
Gambar 3.40. *List Warehouse Order* menampilkan pesanan dengan status berbeda dan opsi aksi

J.3.2 Tombol Approve dan Mekanisme Approval

Lokasi induk dapat melakukan *approval* dengan mengklik tombol *approve* (berwarna hijau) pada pesanan dengan status *pending*. Saat tombol *approve* diklik, sistem memverifikasi ketersediaan stok di lokasi induk dan melakukan persetujuan otomatis jika stok tersedia. Proses *approval* mencakup:

1. *Stock Verification*: Sistem memverifikasi bahwa jumlah stok yang diminta tersedia di lokasi induk dengan kategori dan tipe yang sesuai
2. *Status Update*: Jika stok tersedia, status pesanan berubah dari *pending* menjadi *approved*
3. *Create Shipment Option*: Setelah pesanan diapprove, tombol *create shipment* menjadi aktif untuk memulai proses pengiriman
4. *Rejection Option*: Jika stok tidak tersedia, lokasi induk dapat menolak pesanan dengan mengklik tombol *reject*

Tampilan pesanan setelah *approval* dapat dilihat pada Gambar3.41.



Gambar 3.41. *Warehouse Order* setelah *approval* dengan status berubah menjadi *approved* dan tombol *create shipment* aktif

J.4 Proses Pembuatan Shipment dari Warehouse Order

Setelah *Warehouse Order* diapprove, langkah selanjutnya adalah membuat *shipment* untuk memulai proses pengiriman stok dari lokasi induk ke lokasi pemesan.

J.4.1 Langkah 1: Akses Modul Create Shipment

Lokasi induk mengklik tombol *create shipment* pada pesanan yang sudah diapprove. Sistem akan membuka modul *create shipment* yang menampilkan informasi pesanan dan meminta data pengiriman yang diperlukan:

– *Shipment Information*:

- *Order Reference*: Referensi pesanan dari *Warehouse Order* yang akan dikirim (otomatis terisi)
- *From Location*: Lokasi pengirim (otomatis terisi, contoh: *GMA*)
- *To Location*: Lokasi penerima (otomatis terisi, contoh: *GDB*)
- *SKU*: Produk yang dikirim (otomatis terisi)
- *Units to Ship*: Jumlah unit yang dikirim (dapat disesuaikan jika perlu pengiriman sebagian)
- *Shipping Date*: Tanggal pengiriman ditentukan saat membuat *shipment*
- *Shipment Notes*: *Field* opsional untuk menambahkan catatan pengiriman

J.4.2 Langkah 2: Konfirmasi dan Pembuatan Shipment

Setelah mengisi informasi pengiriman, lokasi induk mengklik tombol *create* untuk membuat *shipment*. Sistem akan memproses data dan menghasilkan:

- *Shipment Code*: Kode pengiriman unik untuk pelacakan (*tracking*)
- *Tracking Reference*: Nomor referensi untuk melacak stok selama dalam perjalanan
- Status *Shipment*: Awalnya *in transit* untuk menunjukkan barang dalam perjalanan

J.5 Penerimaan Warehouse Order di Lokasi Tujuan

Setelah *shipment* dibuat dan barang dikirim, lokasi tujuan akan menerima notifikasi dan dapat melakukan penerimaan stok.

J.5.1 Konfirmasi Penerimaan

Lokasi tujuan menerima *shipment* dengan melakukan verifikasi fisik dan konfirmasi di sistem:

- *Receive Shipment*: Lokasi tujuan mengklik tombol *receive* untuk mengkonfirmasi penerimaan stok
- *Verification*: Sistem memverifikasi jumlah stok yang diterima sesuai dengan *shipment*
- *Status Update*: Status pesanan berubah menjadi *fulfilled* untuk menunjukkan pesanan telah selesai
- *Create Batch*: Stok yang diterima dicatat sebagai *batch* baru di lokasi tujuan untuk inventaris lokal

J.6 Status dan Alur Warehouse Order

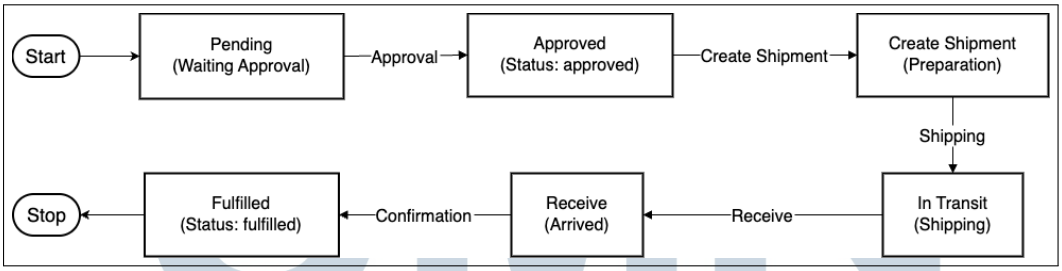
Warehouse Order memiliki beberapa status utama yang menunjukkan tahapan pesanan dalam prosesnya:

J.6.1 Status dalam Lifecycle Warehouse Order

1. *Pending*: Pesanan baru telah dibuat oleh lokasi peminta namun belum mendapat *approval* dari lokasi induk. Pada status ini, pesanan menunggu review dan *approval* stok dari lokasi induk
2. *Approved*: Pesanan telah disetujui oleh lokasi induk dengan verifikasi stok sudah tersimpan. Status ini menandakan bahwa stok siap untuk dikirim dan proses pengiriman dapat dimulai
3. *In Transit*: Pesanan sedang dalam perjalanan dari lokasi induk ke lokasi penerima dengan *shipment code* untuk pelacakan real-time
4. *Fulfilled*: Pesanan telah diterima di lokasi tujuan dan stok telah dikonfirmasi masuk ke inventaris lokal. Status ini menunjukkan pesanan telah selesai dengan sukses

J.7 Alur Lengkap Warehouse Order

Berikut adalah alur lengkap *Warehouse Order* dari pembuatan pesanan hingga penerimaan di lokasi tujuan, seperti yang ditampilkan pada Gambar 3.42:



Gambar 3.42. Alur lengkap *Warehouse Order* dari pembuatan pesanan dengan status *pending* hingga penerimaan dengan status *fulfilled*

Alur ini menunjukkan bagaimana *Warehouse Order* dimulai dengan status *pending* menunggu approval dari parent location, setelah di-approve maka shipment dibuat untuk persiapan pengiriman, barang dalam perjalanan dengan status *in transit*, barang diterima di lokasi tujuan, dan akhirnya *Warehouse Order* ditandai sebagai *fulfilled* setelah konfirmasi penerimaan lengkap.

J.8 Keuntungan Sistem Warehouse Order

Sistem *Warehouse Order* memberikan beberapa keuntungan operasional yang signifikan bagi semua tingkatan hierarki distribusi:

J.8.1 Keuntungan Implementasi

- *Centralized Control*: Semua pesanan tercatat terpusat dengan *audit trail* lengkap untuk keperluan *audit* dan *accountability*
- *Real-Time Tracking*: Setiap pesanan dapat dilacak secara *real-time* dari pembuatan hingga penerimaan dengan *shipment code*
- *Stock Visibility*: Lokasi induk memiliki visibilitas penuh tentang kebutuhan stok di lokasi cabang untuk perencanaan distribusi yang lebih baik
- *Approval Mechanism*: Mekanisme *approval* memastikan stok tidak *overallocated* dan distribusi berjalan terorganisir
- *Automated Batch Creation*: Stok yang diterima otomatis dicatat sebagai *batch* baru untuk inventaris lokal yang terstruktur
- *Universal Application*: Alur yang sama berlaku untuk semua jalur distribusi, memastikan konsistensi dan kemudahan pelatihan pengguna

J.9 Integrasi dengan Sistem Lainnya

Warehouse Order terintegrasi dengan berbagai modul sistem lainnya untuk menciptakan workflow yang seamless:

- *Stock Distribution*: Stok yang diterima melalui *Warehouse Order* dapat segera dikategorisasi melalui *Stock Distribution*
- *Shipment Management*: *Shipment* otomatis dibuat dari *Warehouse Order* yang di-*approve* untuk efisiensi proses
- *Inventory Management*: Stok di lokasi induk berkurang saat *Warehouse Order* di-*approve*, dan stok di lokasi tujuan bertambah saat penerimaan dikonfirmasi
- *Reporting*: Semua transaksi *Warehouse Order* tercatat untuk pelaporan dan analisis distribusi

K Shipment (Pengiriman)

Shipment adalah proses pengiriman barang dari lokasi sumber ke lokasi tujuan yang mencakup pelacakan detail, pengemasan, dan konfirmasi penerimaan. Sistem *shipment* dirancang untuk memberikan visibilitas penuh tentang alur barang selama dalam perjalanan dengan mekanisme *tracking* yang akurat dan transparan. Setiap *shipment* memiliki kode unik dan informasi detail untuk memastikan barang sampai ke lokasi tujuan dengan aman dan tepat waktu.

K.1 Peran dan Fungsi Shipment

Shipment memiliki peran krusial dalam sistem distribusi karena menghubungkan lokasi sumber dengan lokasi tujuan melalui proses pengiriman yang terstruktur. *Shipment* memungkinkan *tracking real-time* dan dokumentasi lengkap untuk setiap pengiriman barang dalam jaringan distribusi.

K.1.1 Alur Universal Shipment

Alur *shipment* berlaku secara universal untuk semua *Warehouse Order* yang di-*approve* dalam sistem:

- Dibuat dari *Warehouse Order* yang sudah di-*approve* oleh lokasi induk
- Menggabungkan satu atau lebih *order* dari lokasi pemesan yang sama
- Memiliki *tracking code* unik untuk pelacakan selama dalam perjalanan
- Mencakup detail pengemasan (*packaging*) dengan label dan nomor referensi
- Menghasilkan konfirmasi penerimaan saat barang tiba di lokasi tujuan

K.2 Proses Pembuatan Shipment

Proses pembuatan *shipment* dimulai dari lokasi sumber yang membuat *shipment* berdasarkan *Warehouse Order* yang sudah diapprove. Proses ini mencakup beberapa langkah terstruktur untuk memastikan pengiriman berjalan dengan baik.

K.2.1 Langkah 1: Akses Modul Create Shipment

Lokasi sumber membuka modul *create shipment* di sistem setelah *Warehouse Order* diapprove. Pada tampilan *form*, lokasi diminta untuk mengisi informasi pengiriman dengan detail komprehensif. *Form* ini menampilkan beberapa *section* utama:

– *Combine Orders Section*:

- *Combine Orders*: *Field* untuk memilih satu atau lebih *Warehouse Order* yang akan digabung dalam satu *shipment* (contoh: *MGS-FN-OG-GMA-29K-251218-1635 (GDB+GMA)*)
- *Instructions*: Instruksi untuk memilih *order* dari lokasi tujuan yang sama (contoh: "Pilih beberapa OG (semua harus dari lokasi asal yang sama)")
- *Source OGs*: Menampilkan kode pesanan yang dipilih (contoh: *MGS-FN-OG-GMA-29K-251218-1635*)

– *Shipment Details Section*:

- *Destination (Child)*: Lokasi tujuan pengiriman (otomatis terisi dari *order*, contoh: Gudang Distributor (*GDB*))
- *Total Units*: Jumlah total unit yang akan dikirim (contoh: 29000)
- *Order Date*: Tanggal *order* dibuat (contoh: 18/12/2025)
- *Shipped Date*: Tanggal pengiriman ditentukan saat membuat *shipment* (contoh: 19/12/2025)
- *Shipped Time*: Waktu pengiriman (contoh: 04.57)
- *Expired Date (all items)*: Tanggal kadaluarsa barang dalam *shipment* (contoh: 16/11/2029) dengan catatan bahwa semua *item* dalam *shipment* akan memiliki tanggal *expired* yang sama

– *Logistics Section*:

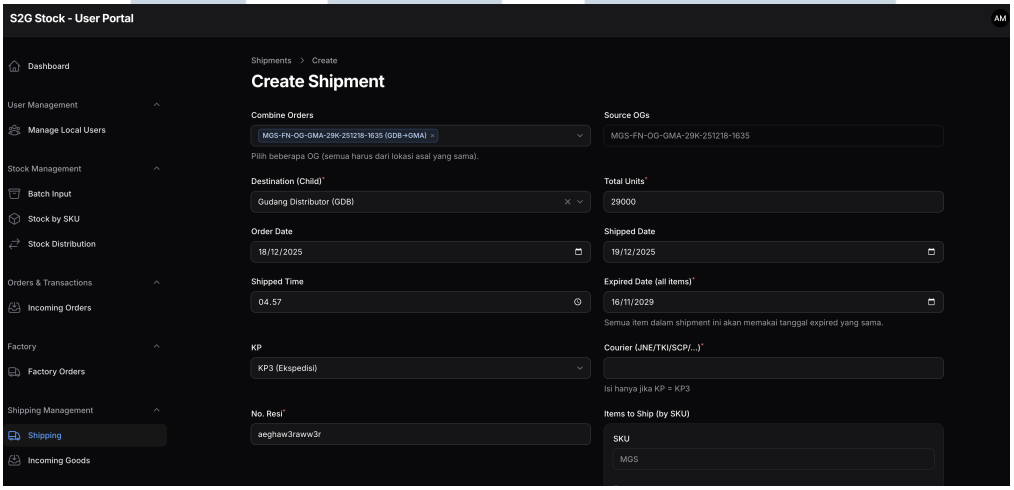
- *KP* (*Kurir Pengiriman*): Jenis kurir pengiriman yang digunakan (contoh: *KP3* (Ekspedisi)) dengan opsi *dropdown*
- *Courier (JNE/TKI/SCP/...)*: Nama perusahaan kurir yang ditentukan (contoh: *JNT* untuk *JNT Express*)

- *No. Resi*: Nomor resi/referensi pengiriman dari kurir (contoh: *aeghaw3raww3r*)

– *Items to Ship Section*:

- *SKU*: Kode produk yang akan dikirim (otomatis terisi, contoh: *MGS*)
- Menampilkan detail *item* dengan kategori dan sub-kategori yang akan dikirim

Tampilan *form create shipment* dapat dilihat pada Gambar 3.43.



Gambar 3.43. *Form create Shipment* untuk mengatur pengiriman dengan detail kurir dan pengemasan

K.2.2 Langkah 2: Pengemasan dan Package Breakdown

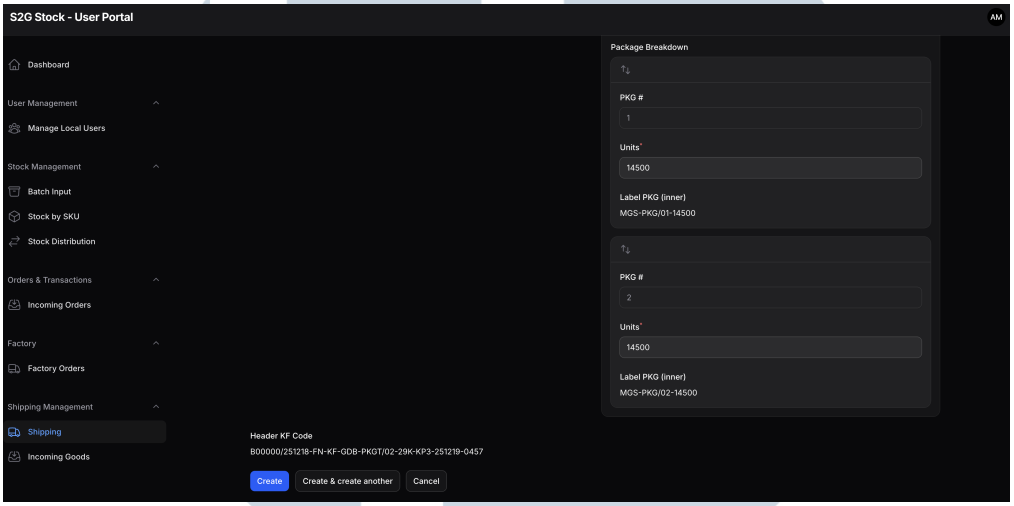
Setelah mengisi informasi pengiriman dasar, sistem menampilkan *section Package Breakdown* yang mendetail tentang pengemasan barang. Section ini menampilkan bagaimana barang akan diatur dalam *packages* untuk pengiriman:

– *Package Breakdown Section*:

- *PKG #*: Nomor paket dalam *shipment* (contoh: 1, 2 untuk *multiple packages*)
- *Units*: Jumlah unit per paket (contoh: 14500 unit per *package*)
- *Label PKG (inner)*: Label identifikasi paket dengan kode unik (contoh: *MGS-PKG/01-14500*, *MGS-PKG/02-14500*)

– *Header KF Code*: Kode *header* untuk *tracking* fisik pengiriman (contoh: *B00000/251218-FN-KF-GDB-PKGT/02-29K-KP3-251219-0457*) yang berfungsi sebagai identitas unik untuk seluruh *shipment*

Tampilan *section package breakdown* dapat dilihat pada Gambar 3.44.



Gambar 3.44. Section *Package Breakdown* menampilkan detail pengemasan dengan nomor paket dan label identifikasi

K.2.3 Langkah 3: Konfirmasi dan Pembuatan Shipment

Setelah mengisi semua informasi pengiriman dan pengemasan, lokasi sumber memiliki tiga opsi aksi:

1. *Create*: Membuat *shipment* dan menyimpannya ke sistem dengan status *on_shipping* untuk menunjukkan barang sedang dalam perjalanan (tombol berwarna biru)
2. *Create & create another*: Membuat *shipment* dan membuka form baru untuk pengiriman tambahan
3. *Cancel*: Membatalkan proses pembuatan *shipment*

Saat mengklik *create*, sistem akan memproses data dan menghasilkan:

– *Shipment Code*: Kode pengiriman unik untuk *tracking* (contoh: *SHP-GDB-251219-0457*)

– *Header KF Code*: Kode fisik untuk label pengiriman yang dicetak

– *Status*: Awalnya *on_shipping* untuk menunjukkan barang dalam perjalanan

– *Tracking Reference*: Nomor referensi untuk pelacakan *real-time* selama pengiriman

K.3 Tampilan Detail Shipment

Setelah *shipment* dibuat, lokasi dapat melihat detail lengkap tentang pengiriman yang mencakup informasi komprehensif tentang barang yang dikirim.

K.3.1 View Shipment - Combine Orders Section

Tampilan detail *shipment* menampilkan dua *section* utama yang memberikan informasi lengkap tentang pengiriman:

– *Combine Orders Section*:

- *Source OGs*: Kode pesanan yang digabung dalam *shipment* (contoh: *MGS-FN-OG-GMA-29K-251218-1635*)
- *Destination (Child)*: Lokasi tujuan pengiriman (contoh: *GDB*)
- *Total Units*: Jumlah unit dalam *shipment* (contoh: 29,000)
- *Order Date*: Tanggal *order* dibuat (contoh: 18 Dec 2025)
- *Status*: Status pengiriman saat ini (contoh: *on_shipping* dengan *badge visual indicator*)

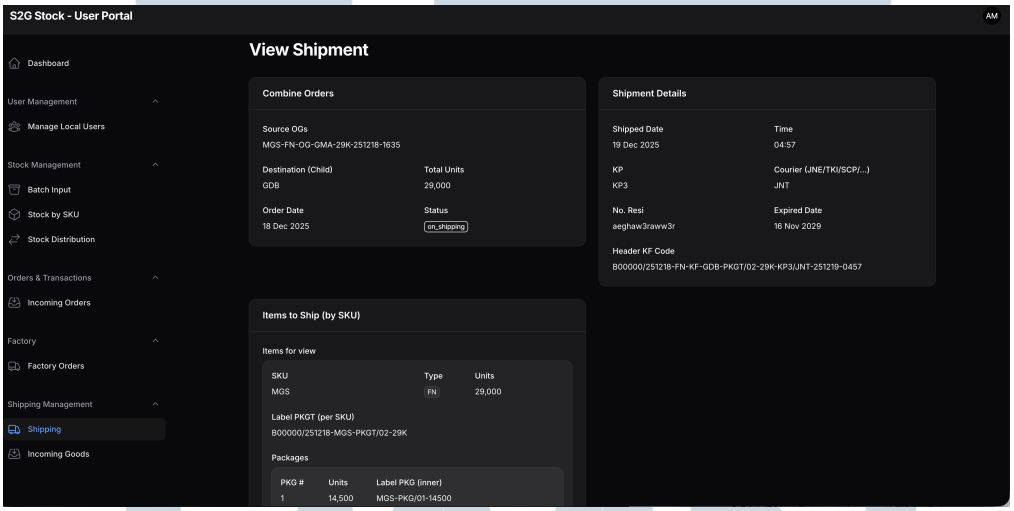
– *Shipment Details Section*:

- *Shipped Date*: Tanggal barang dikirim (contoh: 19 Dec 2025)
- *Time*: Waktu pengiriman (contoh: 04:57)
- *KP*: Jenis kurir pengiriman (contoh: *KP3*)
- *Courier (JNE/TKI/SCP/...)*: Nama kurir (contoh: *JNT*)
- *No. Resi*: Nomor resi dari kurir (contoh: *aeghaw3raww3r*)
- *Expired Date*: Tanggal kadaluarsa barang (contoh: 16 Nov 2029)
- *Header KF Code*: Kode identifikasi fisik pengiriman (contoh: *B00000/251218-FN-KF-GDB-PKGT/02-29K-KP3/JNT-251219-0457*)

– *Items to Ship (by SKU) Section*:

- *Items for view*: Menampilkan detail *SKU* yang dikirim
- *Label PKGT*: Label pengiriman untuk *tracking* (contoh: *B00000/251218-MGS-PKGT/02-29K*)
- *Packages*: Detail paket dengan informasi pengemasan:
  - *PKG #*: Nomor paket (contoh: 1)
  - *Units*: Unit per paket (contoh: 14,500)
  - *Label PKG (inner)*: Label identifikasi paket (contoh: *MGS-PKG/01-14500*)

Tampilan detail *shipment* dapat dilihat pada Gambar3.45.



Gambar 3.45. *View Shipment* menampilkan detail pengiriman lengkap dengan kombinasi pesanan dan detail pengemasan

K.4 Penerimaan Shipment di Lokasi Tujuan

Setelah *shipment* dibuat dan barang dalam perjalanan, lokasi tujuan akan menerima notifikasi dan dapat melakukan penerimaan barang.

K.4.1 Proses Receive Shipment

Lokasi tujuan menerima *shipment* dengan melakukan verifikasi fisik dan konfirmasi di sistem:

– *Verification*: Lokasi tujuan melakukan verifikasi fisik barang yang diterima untuk memastikan:

- Jumlah unit sesuai dengan *shipment* (contoh: 29,000 unit)
- Kondisi barang sesuai ekspektasi
- Label dan paket teridentifikasi dengan jelas
- Tidak ada kerusakan atau kehilangan selama pengiriman

– *Confirmation in System*: Lokasi tujuan mengklik tombol *receive* untuk mengkonfirmasi penerimaan:

- *Status Update*: Status *shipment* berubah dari *on_shipping* menjadi *received*
- *Create Batch*: Sistem otomatis membuat *batch* baru di lokasi tujuan dengan kode *batch* yang terstruktur
- *Timestamp*: Dicatat waktu penerimaan untuk *audit trail*

– *Batch Creation*: Stok yang diterima dicatat sebagai *batch* baru di lokasi tujuan:

- *Batch Code*: Kode *batch* baru yang di-*generate* dari *shipment reference*
- *Source*: Lokasi sumber pengiriman (contoh: *GMA*)
- *Target*: Lokasi tujuan penerimaan (contoh: *GDB*)
- *Items*: Semua *item* dari *shipment* dicatat sebagai *stock items* di lokasi tujuan
- *Status*: *Batch* siap untuk dikategorisasi atau didistribusikan lebih lanjut

K.5 Status dan Alur Shipment

Shipment memiliki beberapa status utama yang menunjukkan tahapan pengiriman:

K.5.1 Status dalam Lifecycle Shipment

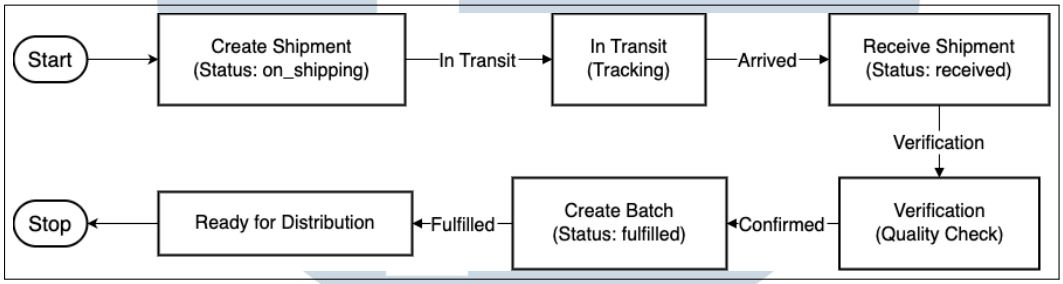
1. *Pending*: Pesanan gudang di-*approve* namun belum dilakukan pengiriman (state awal sebelum membuat *shipment*)
2. *On Shipping* (*In Transit*): *Shipment* telah dibuat dan barang sedang dalam perjalanan dari lokasi sumber ke lokasi tujuan dengan *tracking code* untuk pelacakan *real-time*

3. *Received*: *Shipment* telah sampai di lokasi tujuan dan dikonfirmasi penerimaan oleh lokasi tujuan. Status ini menandakan barang siap untuk dikategorisasi atau didistribusikan lebih lanjut

4. *Fulfilled*: Proses pengiriman dan penerimaan telah selesai dengan sempurna dan *batch* baru telah diciptakan di lokasi tujuan

K.6 Alur Lengkap Shipment

Berikut adalah alur lengkap *shipment* dari pembuatan hingga penerimaan dan pembuatan *batch* baru, seperti yang ditampilkan pada Gambar 3.46:



Gambar 3.46. Alur lengkap *shipment* dari pembuatan pesanan pengiriman hingga pembuatan *batch* baru di lokasi tujuan

Alur ini menunjukkan bagaimana *shipment* dibuat dengan status awal *on_shipping*, kemudian barang dalam perjalanan dengan *tracking real-time*, diterima di lokasi tujuan dengan status *received*, diverifikasi melalui *quality check*, selanjutnya sistem otomatis membuat *batch* baru dengan status *fulfilled*, dan akhirnya *batch* siap untuk didistribusikan ke end point sesuai dengan tujuan yang telah ditentukan.

K.7 Keuntungan Sistem Shipment

Sistem *shipment* memberikan beberapa keuntungan operasional yang signifikan bagi semua tingkatan hierarki distribusi:

K.7.1 Keuntungan Implementasi

- *Real-Time Tracking*: Setiap *shipment* dapat dilacak secara *real-time* dengan *shipment code* dan *tracking reference* untuk visibilitas penuh

- *Combine Orders*: Kemampuan menggabungkan *multiple orders* dalam satu *shipment* untuk efisiensi logistik

– *Detailed Packaging*: Informasi detail tentang pengemasan dengan label dan nomor referensi untuk identifikasi yang akurat

– *Audit Trail*: Setiap *shipment* tercatat dengan *timestamp* lengkap untuk keperluan *audit* dan *accountability*

– *Automated Batch Creation*: *Batch* baru otomatis dibuat di lokasi tujuan saat *shipment* diterima

– *Multi-Courier Support*: Mendukung berbagai jenis kurir pengiriman untuk fleksibilitas logistik

– *Document Generation*: *Header KF Code* dapat dicetak sebagai label fisik untuk pengiriman

K.8 Integrasi dengan Sistem Lainnya

Shipment terintegrasi dengan berbagai modul sistem lainnya untuk menciptakan *workflow* yang *seamless*:

– *Warehouse Order*: *Shipment* dibuat dari *Warehouse Order* yang di-*approve* untuk efisiensi proses

– *Stock Batch*: *Shipment* yang diterima otomatis membuat *batch* baru di lokasi tujuan

– *Stock Distribution*: *Batch* dari *shipment* dapat segera dikategorisasi melalui *Stock Distribution*

– *Reporting*: Semua transaksi *shipment* tercatat untuk pelaporan dan analisis logistik

– *Inventory Management*: Stok di lokasi sumber berkurang saat *shipment* dibuat, dan stok di lokasi tujuan bertambah saat penerimaan dikonfirmasi

L Incoming Goods (Barang Masuk)

Incoming Goods adalah proses penerimaan barang yang tiba dari *shipment* pengiriman dengan verifikasi fisik dan pencatatan dalam sistem. Modul ini merupakan tahap akhir dari alur distribusi di mana lokasi tujuan menerima barang

yang dikirim dari lokasi sumber dan membuat *batch* baru untuk inventaris lokal. Sistem *Incoming Goods* dirancang untuk memastikan setiap barang yang masuk tercatat dengan akurat dan siap untuk didistribusikan lebih lanjut atau digunakan sesuai kebutuhan.

L.1 Peran dan Fungsi Incoming Goods

Incoming Goods memiliki peran krusial dalam sistem distribusi karena menghubungkan proses pengiriman dengan pencatatan stok lokal. *Incoming Goods* memungkinkan lokasi tujuan untuk menerima, memverifikasi, dan mencatat barang yang masuk dengan transparansi penuh dan *audit trail* yang lengkap.

L.1.1 Alur Universal Incoming Goods

Alur *Incoming Goods* berlaku secara universal untuk semua *shipment* yang tiba di lokasi tujuan:

- Sistem menampilkan daftar *shipment* yang sedang dalam status *on_shipping* (dalam perjalanan)
- Lokasi tujuan melakukan penerimaan melalui tombol *receive* untuk *shipment* yang sudah tiba
- Sistem otomatis membuat *batch* baru berdasarkan informasi *shipment*
- *Batch* yang baru dibuat siap untuk dikategorisasi atau didistribusikan lebih lanjut
- Proses dicatat dalam *audit trail* untuk keperluan *tracking* dan *reporting*

L.2 Tampilan Incoming Goods List

Incoming Goods menampilkan daftar semua *shipment* yang sedang dalam perjalanan atau menunggu penerimaan di lokasi tujuan. Tampilan ini memberikan visibilitas penuh tentang barang yang akan tiba dan status penerimaannya.

L.2.1 List Incoming Goods

Daftar *Incoming Goods* menampilkan informasi ringkas tentang setiap *shipment* yang akan diterima dengan fitur *filtering* dan pencarian:

– *Search Bar*: *Field* pencarian untuk menemukan *shipment* berdasarkan *KF Code* atau informasi lainnya

– *Table Columns* (Informasi *Shipment*):

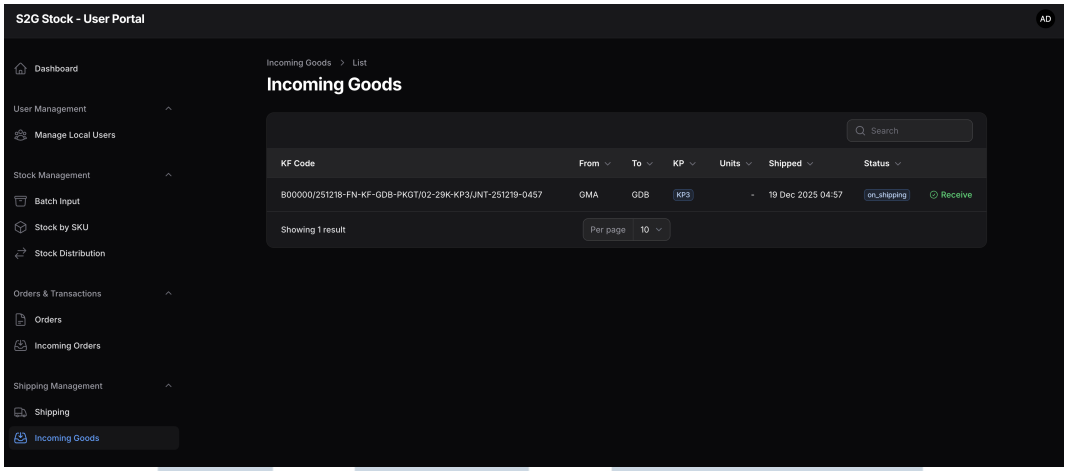
- *KF Code*: Kode identifikasi fisik pengiriman dari *shipment* (contoh: *B00000/251218-FN-KF-GDB-PKGT/02-29K-KP3/JNT-251219-0457*)
- *From*: Lokasi sumber pengiriman (contoh: *GMA*)
- *To*: Lokasi tujuan penerimaan (contoh: *GDB*)
- *KP*: Jenis kurir pengiriman (contoh: *KP3*)
- *Units*: Jumlah unit dalam *shipment* (contoh: - untuk belum dikonfirmasi)
- *Shipped*: Waktu pengiriman dengan tanggal dan jam (contoh: 19 Dec 2025 04:57)
- *Status*: Status *shipment* saat ini dengan *visual indicator* (contoh: *on_shipping* dengan *badge* berwarna)

– *Action Buttons*:

- *Receive*: Tombol hijau untuk menerima *shipment* yang sudah tiba (muncul untuk *shipment* dengan status *on_shipping*)

– *Pagination*: *Per page dropdown* untuk mengatur jumlah *record* per halaman (contoh: 10 *record*)

Tampilan list *Incoming Goods* dapat dilihat pada Gambar 3.47.



Gambar 3.47. *List Incoming Goods* menampilkan *shipment* yang akan diterima dengan opsi penerimaan

L.3 Proses Penerimaan Barang

Proses penerimaan barang melalui *Incoming Goods* mencakup beberapa tahap untuk memastikan verifikasi dan pencatatan yang akurat.

L.3.1 Langkah 1: Verifikasi Fisik Barang

Sebelum mengklik tombol *receive* di sistem, lokasi tujuan harus melakukan verifikasi fisik barang yang tiba:

– *Physical Verification*:

- Menerima paket dengan *KF Code* yang sesuai dengan daftar *Incoming Goods*
- Menghitung jumlah unit yang diterima untuk memastikan sesuai dengan *shipment* (contoh: 29,000 unit)
- Memeriksa kondisi fisik barang untuk memastikan tidak ada kerusakan atau kehilangan
- Memverifikasi label dan paket (*packages*) tercatat dengan jelas
- Melakukan *checking* terhadap *expiry date* produk untuk memastikan kualitas

L.3.2 Langkah 2: Konfirmasi Penerimaan di Sistem

Setelah verifikasi fisik selesai, lokasi tujuan mengklik tombol *receive* untuk mengkonfirmasi penerimaan di sistem:

- *Click Receive Button*: Mengklik tombol *receive* yang tersedia di daftar *Incoming Goods*
- *System Processing*:
  - Sistem melakukan validasi data *shipment* dan verifikasi ketersediaan stok
  - Status *shipment* berubah dari *on_shipping* menjadi *received*
  - *Timestamp* penerimaan dicatat untuk *audit trail*
  - Sistem otomatis membuat *batch* baru berdasarkan informasi *shipment*
- *Batch Creation*: Setelah penerimaan dikonfirmasi, sistem membuat *batch* baru:
  - *Batch Code*: Kode *batch* baru yang di-*generate* berdasarkan *shipment reference* (contoh: *B00000/251218-TM-KF-GDB-PKGT/02-29K-KP3/JNT-251219-0457*)
  - *Source*: Lokasi sumber pengiriman (contoh: *GMA*)
  - *Target*: Lokasi penerimaan lokal (contoh: *GDB*)
  - *SKU*: Produk yang diterima (contoh: *MGS* untuk Monster Glow Serum)
  - *Product*: Nama lengkap produk (contoh: Monster Glow Serum)
  - *Type*: Tipe produk yang diterima (*FN*, *BT*, *BX*)
  - *Units*: Jumlah unit yang diterima (contoh: 29,000)
  - *Expired Date*: Tanggal kadaluarsa *batch* yang ditentukan saat pembuatan *shipment* (contoh: *E3Y10M* untuk 3 tahun 10 bulan)

L.4 Tampilan Stock Batches

Setelah penerimaan *Incoming Goods* selesai, *batch* baru akan muncul di modul *Stock Batches* sebagai bagian dari inventaris lokal. Tampilan ini menunjukkan semua *batch* yang tersedia di lokasi dengan informasi detail tentang kondisi dan ketersediaan stok.

L.4.1 Stock Batches List

Daftar *Stock Batches* menampilkan informasi lengkap tentang semua *batch* yang tersimpan di lokasi dengan fitur *filtering* dan *bulk actions*:

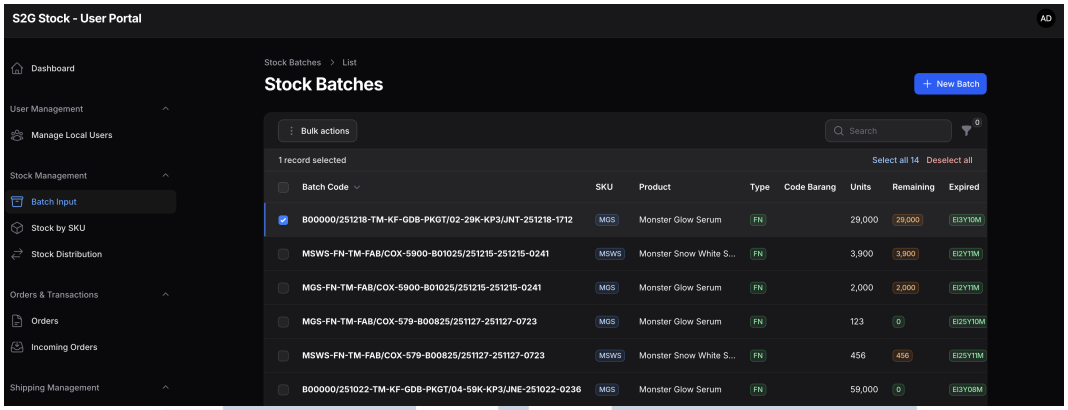
– *Bulk Actions*: Opsi untuk melakukan tindakan massal pada *multiple batches*:
  - *Checkbox* untuk memilih *batch* (*select all* atau *deselect all*)
  - Tombol *Bulk actions* untuk menu tindakan massal yang dapat dilakukan

– *Search Bar*: *Field* pencarian untuk menemukan *batch* berdasarkan *Batch Code*, *SKU*, atau informasi lainnya

– *Table Columns* (Informasi *Batch*):
  - *Batch Code*: Kode *batch* unik (contoh: *B00000/251218-TM-KF-GDB-PKGT/02-29K-KP3/JNT-251218-1712*)
  - *SKU*: Kode produk (contoh: *MGS*, *MSWS*)
  - *Product*: Nama lengkap produk (contoh: Monster Glow Serum, Monster Snow White Serum)
  - *Type*: Tipe produk (*FN* untuk *Final*/Normal, *BT* untuk Botol, *BX* untuk *Box*/Kemasan)
  - *Code Barang*: Kode kategori internal produk
  - *Units*: Jumlah unit dalam *batch* (contoh: 29,000)
  - *Remaining*: Jumlah unit yang masih tersedia untuk didistribusikan (contoh: 29,000 untuk *batch* baru)
  - *Expired*: Informasi kadaluarsa dalam format singkat (contoh: *E3Y10M* untuk 3 tahun 10 bulan, *E2Y1M* untuk 2 tahun 1 bulan)

– *Row Selection*: *Checkbox* pada setiap baris untuk memilih *batch* untuk tindakan massal (contoh: *batch* pertama sudah dipilih)

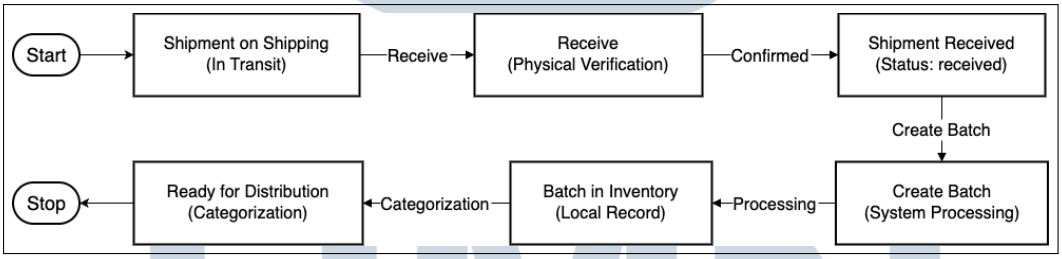
Tampilan *Stock Batches* dapat dilihat pada Gambar 3.48.



Gambar 3.48. *Stock Batches List* menampilkan semua *batch* dengan informasi *detail* dan opsi *bulk actions*

L.5 Alur Lengkap Incoming Goods

Berikut adalah alur lengkap *Incoming Goods* dari penerimaan *shipment* hingga pencatatan*batch* dalam inventaris lokal, seperti yang ditampilkan pada Gambar 3.49:



Gambar 3.49. Alur lengkap *Incoming Goods* dari penerimaan *shipment* hingga pencatatan *batch* dalam inventaris lokal

Alur ini menunjukkan bagaimana *shipment* yang sedang dalam perjalanan diterima di lokasi tujuan melalui verifikasi fisik, *shipment* dicatat dengan status *received*, sistem otomatis membuat *batch* baru berdasarkan data penerimaan, *batch* dicatat dalam inventaris lokal, dan akhirnya *batch* siap untuk didistribusikan atau dialokasikan sesuai dengan kategori dan tujuan penggunaannya.

L.6 Status Incoming Goods

Incoming Goods memiliki status yang menunjukkan tahapan penerimaan barang:

L.6.1 Status dalam Lifecycle Incoming Goods

1. *On Shipping*: *Shipment* sedang dalam perjalanan menuju lokasi tujuan dengan *tracking* melalui *KF Code* dan kurir
2. *Received*: *Shipment* telah tiba di lokasi tujuan dan dikonfirmasi penerimaan melalui sistem dengan *timestamp* lengkap
3. *In Inventory*: Batch baru telah dibuat dan dicatat dalam inventaris lokal, siap untuk dikategorisasi atau didistribusikan lebih lanjut

L.7 Integrasi dengan Sistem Lainnya

Incoming Goods terintegrasi dengan berbagai modul sistem lainnya untuk menciptakan workflow yang seamless dan menutup loop distribusi:

L.7.1 Integrasi Sistem

- *Shipment Management*: *Incoming Goods* menerima data dari *shipment* yang selesai dalam perjalanan
- *Stock Batch*: *Batch* baru otomatis dibuat dari penerimaan *Incoming Goods* dan muncul di modul *Stock Batches*
- *Stock Distribution*: *Batch* dari *Incoming Goods* dapat segera dikategorisasi melalui *Stock Distribution* berdasarkan kondisi fisik
- *Inventory Management*: Stok di lokasi sumber berkurang saat *shipment* dikirim, dan stok di lokasi tujuan bertambah saat *Incoming Goods* diterima
- *Warehouse Order*: *Warehouse Order* ditandai sebagai *fulfilled* setelah *Incoming Goods* selesai diproses
- *Reporting*: Semua transaksi *Incoming Goods* tercatat untuk pelaporan dan analisis distribusi

L.8 Keuntungan Sistem Incoming Goods

Sistem *Incoming Goods* memberikan beberapa keuntungan operasional yang signifikan:

L.8.1 Keuntungan Implementasi

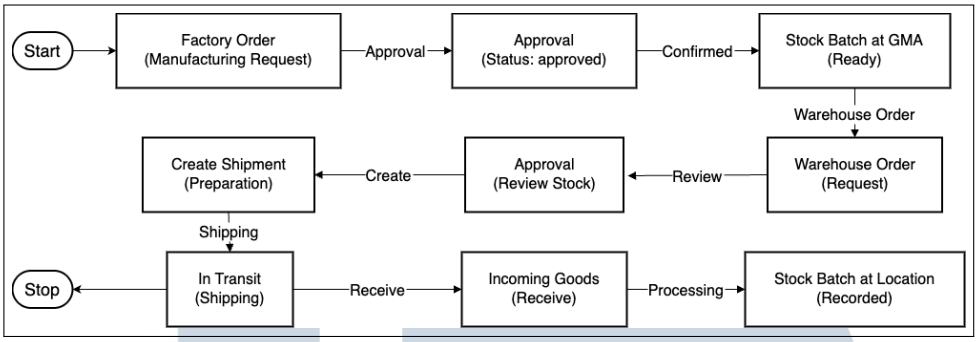
- *Real-Time Visibility*: Lokasi tujuan memiliki visibilitas penuh tentang barang yang akan tiba dengan *KF Code* dan informasi *shipment*
- *Automated Batch Creation*: *Batch* baru otomatis dibuat dari penerimaan *Incoming Goods* tanpa perlu input manual tambahan
- *Accuracy Verification*: Proses verifikasi fisik memastikan hanya barang yang sesuai yang dicatat dalam sistem
- *Audit Trail*: Setiap penerimaan dicatat dengan *timestamp* lengkap untuk keperluan *audit* dan *accountability*
- *Complete Loop Closure*: *Incoming Goods* menutup *loop* distribusi dari pusat manufaktur hingga lokasi tujuan
- *Inventory Synchronization*: Inventaris di semua lokasi tersinkronisasi secara otomatis melalui proses penerimaan
- *Quick Integration*: *Batch* dari *Incoming Goods* siap untuk langsung dikategorisasi atau didistribusikan lebih lanjut

L.9 Loop Distribusi Lengkap

Sistem manajemen stok dalam *S2G Stock WebApp* membentuk loop distribusi yang lengkap dan terstruktur:

L.9.1 Siklus Distribusi End-to-End

Berikut adalah siklus distribusi *end-to-end* yang menunjukkan alur lengkap dari pembuatan *Factory Order* hingga pencatatan *stock batch* di lokasi akhir, seperti yang ditampilkan pada Gambar 3.50:



Gambar 3.50. Siklus distribusi *end-to-end* menunjukkan alur lengkap dari *Factory Order* hingga pencatatan *Stock Batch* di lokasi akhir

Loop ini memastikan bahwa setiap barang dilacak dari pembuatan pesanan manufaktur hingga penerimaan di lokasi akhir dengan transparansi penuh dan kontrol kualitas di setiap tahap. Proses berlangsung secara sistematis: dimulai dengan *Factory Order* yang di-*approve* oleh *GMA*, menghasilkan *Stock Batch* di *GMA*, kemudian lokasi lain melakukan *Warehouse Order* untuk meminta stok, mendapat *approval*, *shipment* dibuat dan dikirim, barang diterima melalui *Incoming Goods*, dan akhirnya dicatat sebagai *Stock Batch* di lokasi tujuan untuk siap didistribusikan lebih lanjut.

M Limitation dan Future Improvement

Meskipun sistem manajemen stok *S2G Stock WebApp* telah dirancang dengan baik, masih terdapat beberapa keterbatasan yang perlu diperbaiki di masa depan. Bagian ini mendokumentasikan keterbatasan yang diketahui dan rencana perbaikan untuk meningkatkan kualitas sistem.

M.1 Keterbatasan Saat Ini

Sistem saat ini masih memiliki beberapa *error* yang mempengaruhi operasional:

- *Units Display Issue*: Kolom *units* pada tampilan *Warehouse Order* dan *Shipment* kadang tidak menampilkan jumlah unit dengan benar, hanya menunjukkan nilai 0 meskipun data sebenarnya ada
- *Stock Deduction Gap*: Sistem belum mengimplementasikan mekanisme pengurangan stok otomatis saat *shipment* dibuat, sehingga stok di lokasi sumber masih terhitung penuh meskipun barang sudah dikirim

– *Header KF Code Accuracy*: Kode *header* pengiriman kadang tidak dihasilkan dengan akurat, sehingga bisa menyulitkan proses pelacakan fisik barang di lapangan

3.4 Kendala dan Solusi

Dalam pelaksanaan kegiatan magang dan pengembangan proyek *website SHiNE2GeTHER*, ditemukan berbagai tantangan yang mempengaruhi proses pengerjaan. Tantangan tersebut mencakup aspek teknis pemrograman, komunikasi manajerial, hingga manajemen waktu dalam lingkungan kerja *startup*. Berikut adalah rincian kendala yang terjadi beserta solusi yang diterapkan.

3.4.1 Kendala

Beberapa hambatan utama yang mempengaruhi alur pengembangan sistem antara lain:

1. Dinamika Komunikasi dan Subjektivitas Desain,
   Terdapat tantangan dalam proses penyelarasan persepsi visual dengan atasan. Kerap terjadi revisi berulang akibat adanya perbedaan preferensi estetika, di mana arahan desain yang diberikan cenderung subjektif dan bergantung pada selera personal, sehingga proses finalisasi antarmuka pengguna (UI) memakan waktu lebih lama dari estimasi.

2. Ketimpangan Distribusi Beban Kerja Teknis,
   Dalam proses kolaborasi tim, teridentifikasi adanya kesenjangan kompetensi teknis (*skill gap*) antar anggota. Hal ini mengakibatkan beban pengerjaan modul-modul krusial dan logika *backend* yang kompleks menjadi tidak seimbang dan cenderung bertumpu pada satu pihak demi menjaga proyek tetap berjalan sesuai target.

3. Interupsi Prioritas dan Tugas tambahan,
   Lingkungan kerja *startup* yang dinamis menuntut fleksibilitas tinggi, sehingga fokus pengerjaan proyek seringkali teralihkan oleh instruksi mendadak. Hal ini meliputi:

   – Pengerjaan proyek lain yang bersifat urgen (seperti manajemen stok).

– Keterlibatan dalam bantuan operasional lapangan (kegiatan di gudang, persiapan *event*, dan logistik tim).

Kondisi ini menyebabkan terpecahnya konsentrasi dan tertundanya *timeline* pengembangan fitur utama.

4. Adaptasi dan Kurva Pembelajaran Teknis
   Implementasi fitur sistem yang kompleks menuntut pemahaman logika pemrograman yang mendalam. Terdapat kebutuhan waktu tambahan untuk mengasah kemampuan *coding* agar dapat memenuhi standar efisiensi, keamanan, dan struktur kode yang layak produksi.

3.4.2 Solusi

Untuk mengatasi kendala-kendala tersebut, diterapkan langkah-langkah penyelesaian sebagai berikut:

1. Pengajuan Desain Berbasis Referensi Komparatif,
   Guna meminimalisir subjektivitas, strategi komunikasi diubah dengan cara mengajukan beberapa opsi desain (*Mockup*) sekaligus. Pengajuan ini disertai dengan referensi dari aplikasi terkemuka sebagai acuan standar industri, sehingga diskusi keputusan desain menjadi lebih objektif dan terarah.

2. Manajemen Waktu Adaptif dan Skala Prioritas,
   Menghadapi beban tugas operasional tambahan, diterapkan manajemen waktu yang lebih ketat melalui penyusunan skala prioritas berdasarkan urgensi. Pemanfaatan waktu di luar jam operasional utama atau di sela-sela tugas lapangan dimaksimalkan untuk mengejar progres pengembangan sistem.

3. Peningkatan Kompetensi Secara Mandiri,
   Kekurangan dalam aspek teknis diatasi melalui pembelajaran mandiri (*self-learning*) yang intensif. Eksplorasi dilakukan dengan mempelajari dokumentasi resmi, forum pengembang, serta menganalisis pola kode (*best practices*) untuk meningkatkan kualitas dan efisiensi penulisan kode.

4. Profesionalisme dan Koordinasi Tim,
   Terkait ketimpangan beban kerja, pendekatan yang diambil adalah membangun komunikasi profesional. Dilakukan koordinasi yang lebih

proaktif untuk membagi tugas-tugas pendukung sesuai kapasitas anggota tim lain, serta tetap menjaga hubungan kerja yang kondusif demi tercapainya tujuan proyek bersama.