

Ruang Lingkup Sistem PFM OCR

Apa yang sudah jalan, apa yang belum

Sistem OCR on-premise ber-GPU yang mengubah foto Surat Jalan dari HP petugas toko menjadi data terstruktur di database. Dokumen ini menjelaskan alurnya, letak tiap bagian di filesystem, port dan jaringannya, cara menjalankan tiap bagian dan dari folder mana, lalu memisahkan dengan tegas mana yang sudah terbangun dan mana yang belum.

Untuk siapa	Orang yang melanjutkan pekerjaan ini, dibantu AI coding agent
Repositori	app-pfm-ocr-v2 (git, branch main)
Disusun dari	isi repo per 27 Agustus 2026 — bukan dari ingatan
Dokumen pendamping	PFM-Scanner-Handover-Document.pdf (42 hal, lebih dalam) dan PFM-Scanner-Knowledge-Transfer.pdf (30 slide)

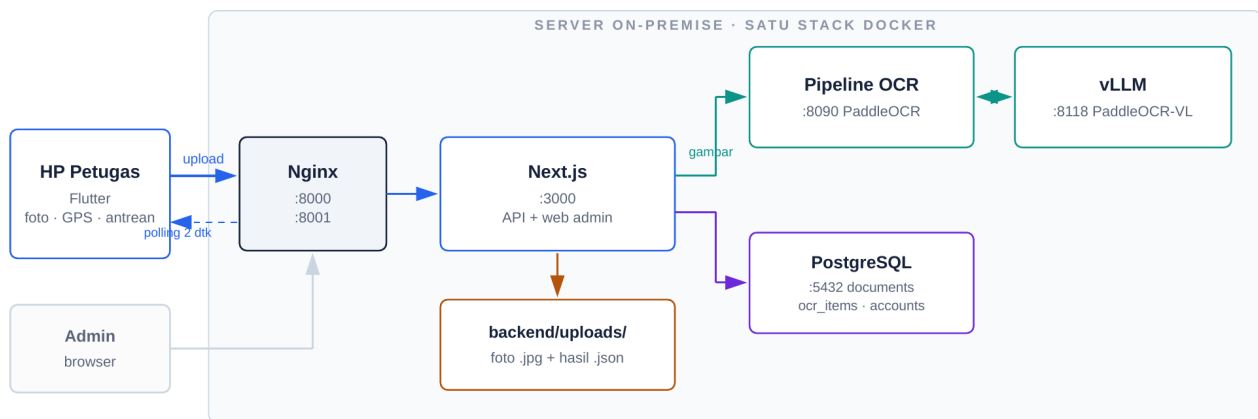
MULAI DARI SINI Baca dokumen ini lebih dulu, baru dua dokumen pendamping di atas. Semua angka, port, nama tabel dan nama file di sini diambil langsung dari file yang di-commit — docker-compose.yml, backend/nginx.conf, app_config.dart, db/init.ts, dan plans/next-enhancements.md. Tidak ada yang dikarang.

1. Sistem ini sebenarnya apa

Petugas toko memotret Surat Jalan (DO) pakai HP. Foto dikirim ke server on-premise ber-GPU, dibaca oleh OCR, hasilnya jadi data terstruktur di database. Petugas mengoreksi hasilnya di HP, lalu mencetak struk. Ada juga web admin untuk melihat riwayat dan mengelola master data.

BAGIAN	LETAK DAN PERANNYA
Aplikasi HP (Flutter)	Akar repo, folder lib/. Dipakai petugas toko. Satu akun terikat ke satu toko.
Web app + API (Next.js)	backend/pfm-web-app/. Otak sistem: melayani aplikasi HP sekaligus jadi halaman admin.
Mesin OCR (Python + GPU)	backend/config/ dan Dockerfile. PaddleOCR + model vision-language vLLM.

Alur besarnya



Semua kotak di dalam bingkai abu-abu adalah container Docker di satu server yang sama.

Delapan langkah, dari foto sampai tersimpan

- 1. Login.** Petugas login. Satu akun terikat ke satu toko. Token diverifikasi sungguhan ke backend, bukan stub.
- 2. Foto DO.** Aplikasi memotret, lalu mengecek ketajaman gambar di HP (Laplacian variance). Kalau buram, ditandai.
- 3. Tempel GPS dan unggah.** Koordinat ikut dikirim ke POST `/api/v1/documents/upload`.
- 4. Antrean tahan mati.** Upload yang belum selesai disimpan di Hive. Aplikasi ter-kill di tengah jalan tidak menghilangkan dokumen — antrean dilanjutkan saat dibuka lagi.
- 5. Server memproses.** Next.js meneruskan gambar ke pipeline OCR di GPU, hasilnya disimpan ke tabel `documents` dan `ocr_items`.
- 6. Aplikasi menunggu.** HP polling tiap 2 detik (maksimal 130 kali, sekitar 4,3 menit) sampai backend menandai `parsed = true`.

7. **Petugas mengoreksi.** Hasil OCR ditampilkan untuk dibetulkan, lalu disimpan lewat PUT `/api/v1/documents/:id`.
8. **Cetak struk.** PDF dibuat langsung di HP dan bisa dicetak di tempat.

2. Peta folder

Satu repo, dua aplikasi. Akar repo adalah aplikasi Flutter; folder backend/ berisi semua yang jalan di server.

Aplikasi Flutter — akar repo

LOKASI	ISINYA
<code>lib/config/</code>	<code>app_config.dart</code> — tema dan alamat server
<code>lib/core/</code>	Dio (HTTP), Hive (simpan lokal), GPS, <code>go_router</code>
<code>lib/features/</code>	satu folder per fitur: auth, camera, documents, editor, splash
<code>lib/data/</code>	<code>master_sku.dart</code> — daftar SKU yang ikut ter-compile ke APK
<code>test/</code>	tes Flutter
<code>android/ ios/ web/</code>	wadah per platform, jarang disentuh

Backend — semua yang jalan di server

LOKASI	ISINYA
<code>backend/pfm-web-app/</code>	Next.js: web admin + API gateway
<code>src/app/api/v1/</code>	API yang dipakai aplikasi HP
<code>src/app/api/</code>	API internal web admin
<code>src/app/admin/</code>	halaman master data
<code>src/db/init.ts</code>	skema + seeding akun (lihat bab 5)
<code>backend/config/</code>	setting pipeline OCR dan vLLM
<code>backend/db/migrations/</code>	skema Postgres, jalan otomatis saat DB masih kosong
<code>backend/uploads/</code>	foto DO masuk + hasil JSON — sekitar 173 MB
<code>backend/sources/</code>	data klien dan ground truth — sekitar 283 MB, RAHASIA
<code>backend/nginx.conf</code>	aturan routing semua port
<code>backend/Dockerfile</code>	tiga target: <code>vllm-server</code> / <code>pipeline-api</code> / <code>pfm-web-app</code>

Di akar repo

LOKASI	ISINYA
<code>docker-compose.yml</code>	stack resmi — jalankan SELALU dari sini
<code>docker-compose.demo.yml</code>	override mode produksi untuk demo

LOKASI	ISINYA
<code>start-dev-tunnel.ps1</code>	sinkronisasi IP LAN ke app_config.dart + ngrok
<code>docs/ plans/</code>	dokumentasi dan backlog
<code>CLAUDE.md AGENTS.md</code>	instruksi untuk AI coding agent

JEBAKAN Ada dua file docker-compose.yml — satu di akar repo, satu di dalam backend/ (sisa lama dengan nama proyek Compose berbeda). Selalu jalankan dari akar repo; menjalankan dari dalam backend/ membuat nama container bentrok dengan yang sudah menyala.

3. Port dan jaringan

Hanya dua port terbuka ke luar server. Sisanya hanya bisa diakses antar-container.

PORT	CONTAINER	ISINYA	AKSES
8000	paddleocr-nginx	Pintu depan. Semua request masuk sini lalu diteruskan.	Publik
8001	paddleocr-nginx	Pintu sempit: hanya <code>/api/v1/*</code> yang lolos, sisanya 404.	Publik
3000	paddleocr-pfm-web-app	Next.js — web admin + API gateway.	Internal
8090	paddleocr-pipeline-api	PaddleOCR pipeline, endpoint <code>/layout-parsing</code> .	Internal
8118	paddleocr-vllm-server	Model vision-language di GPU (PaddleOCR-VL).	Internal
8120	paddleocr-pipeline-api	Classifier <code>/classify-ocr</code> .	Internal
5432	paddleocr-db	Postgres 15, database dopfm.	Internal

Pembagian lalu lintas di Nginx



Diatur di `backend/nginx.conf`. Batas ukuran upload dimatikan (`client_max_body_size 0`) karena foto DO besar; `timeout 300` detik karena OCR lama.

Cara HP menemukan server

Alamat server tidak di-hardcode satu nilai. Setiap aplikasi dibuka, `AppConfig.initializeApiBaseUrl()` mencoba dua alamat berurutan:

- **LAN dulu** — `http://192.168.x.x:8000/api/v1`, cepat, dipakai saat HP dan server satu WiFi.
- **Ngrok kalau LAN gagal** — tunnel publik, dipakai saat petugas di luar jaringan kantor.
- Kalau dua-duanya mati, tetap dipakai alamat LAN supaya pesan error menunjuk ke server yang benar.

Pengecekkannya bukan sekadar “bisa connect”: aplikasi memastikan jawabannya JSON asli dari backend, bukan halaman error tunnel atau halaman login router.

PALING SERING Kalau login atau upload gagal di HP, periksa ini duluan — biasanya IP LAN server berubah atau tunnel ngrok mati. Jalankan `./start-dev-tunnel.ps1`, lalu build ulang APK karena alamatnya konstanta di dalam kode.

4. Cara menjalankan — langkah demi langkah

BACA DULU Aturan folder yang paling sering salah: semua perintah docker compose dan semua perintah flutter dijalankan dari AKAR REPO (app-pfm-ocr-v2/), bukan dari dalam backend/. Perintah npm hanya dari backend/pfm-web-app/, dan itu pun jarang — npm biasanya sudah dijalankan di dalam container.

4.1 Prasyarat di mesin server

BUTUH	KETERANGAN
Docker + Compose v2	Perintahnya docker compose (pakai spasi), bukan docker-compose.
GPU NVIDIA + Container Toolkit	Wajib. Compose meminta driver: nvidia untuk vllm-server dan pipeline-api. Tanpa toolkit, container gagal start.
Flutter SDK 3.2.0+	Hanya di mesin developer, bukan di server. Lihat pubspec.yaml.
ngrok (opsional)	Hanya kalau perlu akses dari luar jaringan kantor.

4.2 Setup pertama kali (sekali saja)

```
cd app-pfm-ocr-v2 # semua dari akar repo

cp backend/.env.example backend/.env

# edit backend/.env — isi tiga nilai ini:
# APP_PORT=8000 port yang dibuka ke luar
# CUDA_VISIBLE_DEVICES=0 nomor GPU yang dipakai (cek: nvidia-smi)
# JWT_SECRET=<rahasia-baru> JANGAN biarkan "change-me"

docker compose up --build # pertama kali LAMA: unduh model + image
```

Saat pertama kali, Postgres menjalankan sendiri semua file di backend/db/migrations/. Ini hanya terjadi kalau volume database masih kosong. Kalau nanti ada migration baru sedangkan database sudah berisi, file itu tidak jalan otomatis — harus dijalankan manual (lihat 4.7).

4.3 Menjalankan backend sehari-hari (mode dev)

```
cd app-pfm-ocr-v2 # WAJIB dari akar repo

docker compose up # jalan di depan, log langsung kelihatan
docker compose up -d # jalan di belakang layar
docker compose up --build # kalau Dockerfile / dependency berubah
```

Mode ini menjalankan npm run dev di dalam container, dengan folder backend/pfm-web-app di-mount masuk. Artinya edit file TypeScript di komputer langsung berubah, tanpa rebuild. Ini yang dipakai untuk

ngoprek parser sehari-hari.

4.4 Menjalankan backend untuk demo ke klien (mode produksi)

```
cd app-pfm-ocr-v2
```

```
docker compose -f docker-compose.yml -f docker-compose.demo.yml up -d --build
```

--build di sini wajib, bukan opsional. Mode demo membuang mount source dan menjalankan npm start terhadap hasil npm run build yang sudah tertanam di image. Konsekuensinya mode ini tidak hot-reload: kalau ada perubahan kode dan Anda lupa --build, yang jalan adalah kode lama dari image sebelumnya.

Kenapa harus mode ini saat demo: dev server hanya satu proses dan akan tercekik kalau beberapa HP mengunggah bersamaan.

4.5 Memastikan backend sudah hidup

```
docker compose ps # semua harus running / healthy
```

```
curl http://localhost:8000/api/v1/health # harus balas JSON status ok
```

```
curl http://localhost:8001/api/v1/health # port khusus aplikasi HP
```

```
docker compose logs -f pfm-web-app # log Next.js (paling sering dibaca)
```

```
docker compose logs -f pipeline-api # log OCR
```

```
docker compose logs -f vllm-server # cek kalau OCR lambat / gagal
```

Urutan hidupnya bertahap: db dan vllm-server dulu, lalu pipeline-api menunggu sampai healthy, baru pfm-web-app jalan. Jadi wajar kalau saat start awal pfm-web-app belum langsung menyala. vllm-server memuat model ke GPU dan ini bisa makan beberapa menit saat pertama kali.

4.6 Menghentikan dan membersihkan

```
docker compose stop # hentikan, data & container tetap ada
```

```
docker compose down # hapus container, DATA TETAP AMAN
```

```
docker compose restart pfm-web-app # restart satu service saja
```

```
docker compose down -v # HAPUS VOLUME — DATABASE IKUT TERHAPUS
```

Jangan asal ketik down -v. Flag -v menghapus volume paddleocr_pgdata — seluruh isi database hilang dan tidak ada backup otomatis. Cache model juga terhapus, jadi model harus diunduh ulang.

4.7 Mengakses database

```
docker compose exec db psql -U postgres -d dopfm # masuk psql
```

```
# di dalam psql: \dt (lihat tabel) \q (keluar)
```

```
# menjalankan migration baru pada DB yang sudah berisi:
docker compose exec -T db psql -U postgres -d dopfm \
< backend/db/migrations/00X_nama.sql
```

4.8 Menjalankan aplikasi Flutter

```
cd app-pfm-ocr-v2 # akar repo, BUKAN folder android/

flutter pub get # setelah clone / ubah pubspec
flutter devices # pastikan HP / emulator terdeteksi
flutter run # jalankan ke perangkat terhubung

flutter test # semua tes
flutter test test/blur_detector_test.dart # satu file saja
flutter analyze lib # cek statis – jalankan sebelum commit
```

DIJALANKAN DI	ALAMAT LAN YANG BENAR DI APP_CONFIG.DART
HP fisik	http://<IP LAN mesin server>:8000/api/v1
Emulator Android	http://10.0.2.2:8000/api/v1
iOS Simulator	http://localhost:8000/api/v1

4.9 Membuat APK untuk dibagikan ke toko

```
cd app-pfm-ocr-v2

# 1. pastikan alamat server di lib/config/app_config.dart sudah BENAR
# (_lanBaseUrl dan _ngrokBaseUrl) – nilainya ikut ter-compile ke APK

flutter clean # kalau hasil build sebelumnya mencurigakan
flutter pub get
flutter build apk --release

# hasilnya: build/app/outputs/flutter-apk/app-release.apk
```

Ini konsekuensi terbesar yang sering terlupa: alamat server adalah konstanta di dalam kode, jadi ikut tertanam di APK. Setiap kali IP LAN server berubah atau domain ngrok berganti, APK harus dibangun ulang dan disebar ulang ke semua HP toko. Ini juga sebabnya “alamat bisa diubah tanpa build ulang” masuk daftar yang belum ada di bab 7c.

APK release saat ini masih ditandatangani kunci debug — cukup untuk instalasi internal, belum bisa masuk Play Store.

4.10 Membuka akses dari luar jaringan (tunnel)

Cara A — ngrok, lewat script (Windows)

```
cd app-pfm-ocr-v2
./start-dev-tunnel.ps1
```

Script ini mengerjakan tiga hal berurutan: mendeteksi IP LAN aktif mesin ini (melewati adapter virtual dan WSL), menuliskan IP itu ke `_lanBaseUrl` di `lib/config/app_config.dart`, lalu menyalakan ngrok. Setelah itu ia menunggu sampai health check LAN dan tunnel dua-duanya menjawab ok.

Perhatikan: ngrok diarahkan ke port 8001 — pintu sempit yang hanya melayani `/api/v1` — bukan 8000. Jadi web admin tidak ikut terbuka ke internet. Setelah script selesai, build ulang APK karena file config baru saja berubah.

Cara B — Cloudflare, tanpa akun

```
cd app-pfm-ocr-v2
docker compose --profile tunnel up -d
docker logs paddleocr-tunnel # cari baris https://xxxx.trycloudflare.com
```

Service tunnel ini mati secara default (pakai profiles), jadi harus dinyalakan sengaja. URL-nya berganti setiap restart, dan mengarah ke port 8000 — semua path, termasuk web admin.

4.11 Bekerja langsung di dalam container web app

```
docker compose exec pfm-web-app sh # masuk shell container
# di dalamnya: npm run lint, npm run build, dsb.
```

Umumnya tidak perlu npm install di komputer sendiri — `node_modules` hidup di dalam container sebagai volume terpisah. Kalau menambah dependency baru, tambahkan ke `package.json` lalu jalankan `docker compose up --build` supaya image dibangun ulang.

5. Alur data — dari mana ke mana

Bab ini menelusuri data yang sebenarnya mengalir, lengkap dengan nama file dan nama tabelnya, supaya bisa langsung dicari di kode.

5.1 Username dan password itu dari mana?

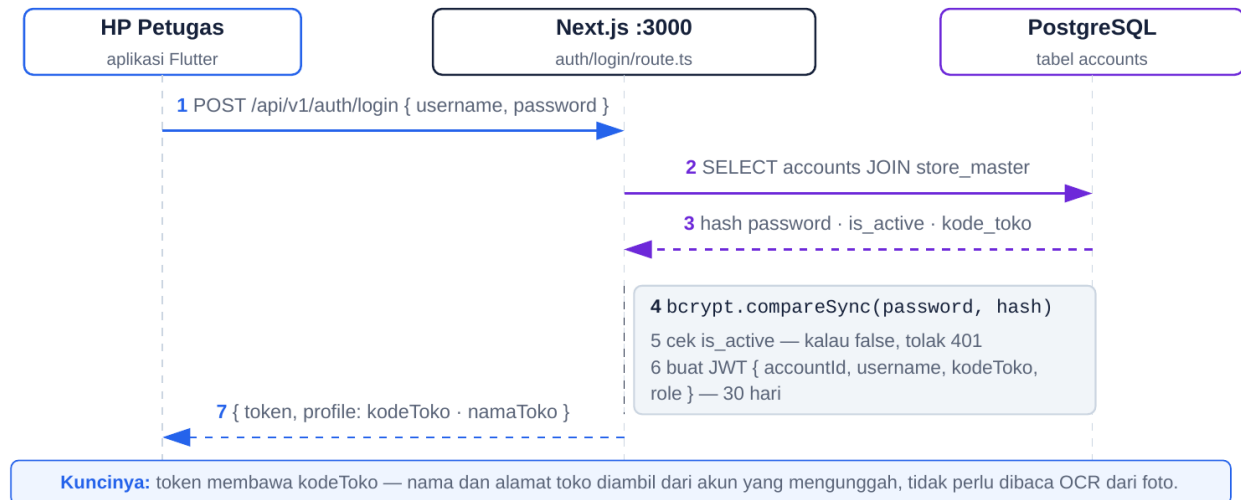
Bukan dari file, melainkan dari **tabel accounts di Postgres**. Akun-akun itu dibuat otomatis oleh sistem, tidak diketik manual. Prosesnya ada di [backend/pfm-web-app/src/db/init.ts](#).

LANGKAH	YANG TERJADI
1. Daftar toko masuk	Tabel <code>store_master</code> diisi dari data master toko (<code>kode_toko</code> , <code>nama_toko</code> , <code>alamat</code>).
2. Akun toko dibuat	Untuk SETIAP baris di <code>store_master</code> (kecuali <code>WH_JOFFICE</code>) dibuatkan satu akun: <code>username</code> = kode toko, <code>password</code> = "password", <code>role</code> = store.
3. Akun admin dibuat	Satu akun admin / password dengan role admin, terikat ke <code>WH_JOFFICE</code> .
4. Password lama dirapikan	Password yang masih polos (tidak diawali \$2) di-hash ulang dengan <code>bcrypt</code> .

Jadi kalau ada toko baru, caranya bukan membuat akun — cukup **tambahkan tokonya ke `store_master`**, akunnya menyusul sendiri.

PENTING Dua hal yang wajib diketahui soal password ini. (1) Password awal semua akun adalah kata "password", sama untuk semua toko. (2) Seeding di atas memakai `ON CONFLICT (username) DO UPDATE SET password`, dan `initDb()` dipanggil setiap kali container web app menyentuh database pertama kali setelah start — artinya SETIAP RESTART BACKEND, password semua akun kembali ke "password", termasuk yang sudah diganti. Belum ada fitur ganti password yang bertahan. Ini perlu dibereskan sebelum dipakai produksi luas.

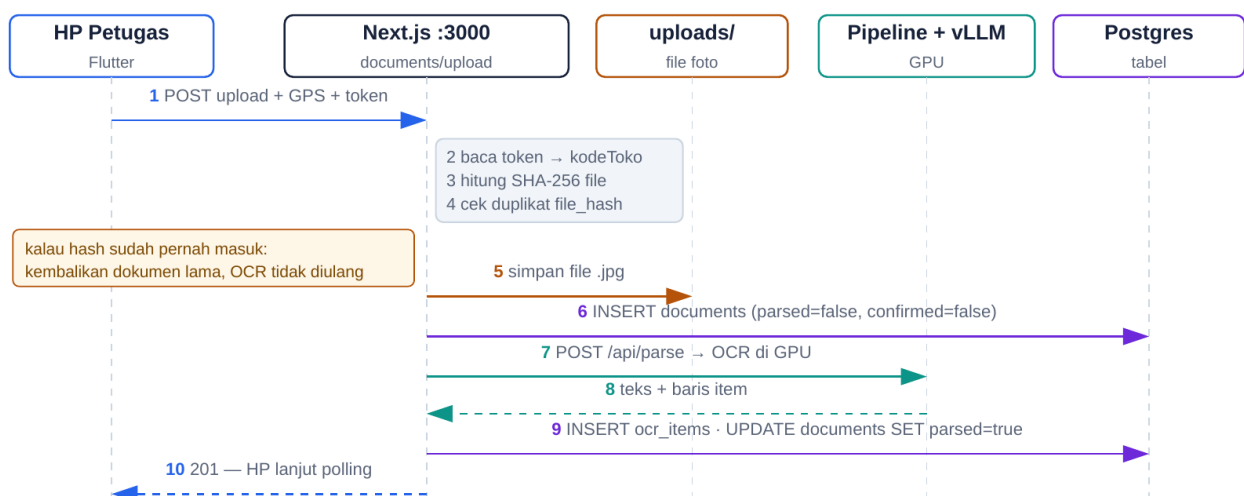
5.2 Apa yang terjadi saat petugas menekan “Login”



Token disimpan di HP lewat Hive. Setiap permintaan berikutnya membawanya di header `Authorization: Bearer <token>`. Saat aplikasi dibuka lagi, `checkLoginState()` memanggil `GET /api/v1/auth/me` untuk memastikan token masih sah; kalau dijawab 401, petugas dikeluarkan.

Perhatikan `JWT_SECRET`. Kalau tidak diisi di `backend/.env`, kode memakai nilai cadangan "dev-only-insecure-secret-change-me" yang tertulis di source. Siapa pun yang tahu nilai itu bisa membuat token palsu untuk toko mana pun. Wajib diisi nilai acak sendiri.

5.3 Apa yang terjadi saat foto diunggah



Tiga hal yang perlu dipahami dari alur ini:

- **Anti-duplikat pakai hash isi file.** Kalau HP mengirim ulang foto yang sama (karena upload

sebelumnya dikira gagal padahal berhasil), server mengenalinya dari SHA-256 dan mengembalikan dokumen lama — tidak membuat baris kedua dan tidak menjalankan OCR ulang. Ini yang membuat “ulangi upload” aman.

- **Foto disimpan sebagai file biasa, bukan di database.** Di dalam container path-nya /uploads, di komputer itu folder backend/uploads/. Yang masuk database hanya nama file-nya. Jadi database dan folder uploads harus di-backup berpasangan — kalau salah satu hilang, yang tersisa tidak berguna.
- **Dokumen baru lahir dengan confirmed = false.** Selama masih false, dokumen itu tidak muncul di GET /api/v1/documents. Ia baru muncul setelah petugas menyimpan hasil koreksi lewat PUT /api/v1/documents/:id. Inilah gerbang konfirmasi.

5.4 Kenapa HP perlu polling

Balasan upload di langkah 10 dikirim dengan field masih kosong, karena OCR bisa memakan waktu lama. Karena itu HP tidak menunggu di satu permintaan, melainkan polling GET /api/v1/documents tiap 2 detik (maksimal 130 kali, sekitar 4,3 menit) sampai dokumen bertanda parsed = true.

Sisi server sudah dijaga berlapis: panggilan ke /api/parse dibatasi 210 detik; kalau pipeline gagal, dokumen tetap ditandai selesai dengan metadata placeholder “Not Found” supaya tidak menggantung selamanya; kalau panggilannya sendiri yang gagal, alasannya dicatat di kolom parse_error supaya HP bisa diberi tahu gagal, bukan disuruh menunggu sia-sia.

5.5 Ringkasan: data mendarat di mana

DATA	DISIMPAN DI	BENTUKNYA
Akun & password	tabel accounts	Password di-hash bcrypt. Dibuat otomatis dari store_master.
Daftar toko	tabel store_master	Sumber kebenaran nama & alamat toko. Menentukan akun apa saja yang ada.
Sesi login	Hive di HP	JWT berlaku 30 hari. Tidak ada sesi tersimpan di server.
Foto DO asli	backend/uploads/	File .jpg, nama diawali timestamp. Sekitar 173 MB saat ini.
Hasil OCR mentah	backend/uploads/	File .jpg.json berpasangan dengan fotonya.
Header dokumen	tabel documents	Nama file, waktu, GPS, kode_toko, file_hash, scan_mode, parsed, confirmed, metadata.
Baris item	tabel ocr_items	Satu baris per item: kode_barang, nama_barang, banyak, jumlah.
Master data	sku_master, vendors, customers	Dikelola lewat halaman admin.
Antrean belum terkirim	Hive di HP	Bertahan meski aplikasi ditutup paksa; dilanjutkan saat dibuka lagi.
Struk PDF	dibuat di HP	Tidak disimpan di server sama sekali.

6. Yang SUDAH ada dan berjalan

Ini bukan rencana — semuanya sudah terpasang di kode dan dipakai.

AREA	YANG SUDAH BERJALAN
Login & sesi	Login diverifikasi sungguhan ke POST /api/v1/auth/login (bukan stub). Sesi JWT disimpan di Hive. Splash mengarahkan otomatis sesuai status sesi. checkLoginState() memanggil GET /auth/me dan logout kalau 401.
Kamera & geotag	Deteksi blur Laplacian dengan badge lolos/gagal sebelum upload. Kunci shutter sampai HP diam (IMU). GPS ditempel baik dari kamera maupun galeri.
Antrean upload	Antrean per-item dengan status uploading/processing/success/error, retry (GPS asli dipertahankan), hapus, dan filter pencarian. Sudah tahan mati: antrean dipersistensi ke Hive, jadi aplikasi ter-kill di tengah upload tidak menghilangkan dokumen.
Editor & struk	Validasi field header (format tanggal, nomor PO/SO), tambah/ubah/hapus item, tanda tangan + konfirmasi penerimaan, dan cetak PDF struk langsung dari HP.
Mode scan	Dua mode — DO dan Product Scan — dengan penanda warna berbeda dan state global.
API gateway	Rute upload/parse/documents CRUD, endpoint status GPU, proxy vLLM, dan tool pelabelan manual.
Pipeline OCR	PaddleOCR + klasifikasi vLLM, cache layout di DB, koreksi pergeseran kolom tabel, normalisasi tanggal, plus harness regresi akurasi. Akurasi saat ini sekitar 89,4% (target 95%).
Database	Postgres 15. Tabel inti: documents, ocr_items, accounts, store_master, sku_master, vendors, customers. Migration jalan otomatis saat volume masih kosong.
DevOps	Compose dev (hot-reload) dan compose demo (mode produksi). Script start-dev-tunnel.ps1 sinkronisasi IP LAN + ngrok. Tunnel Cloudflare opsional.

7. Yang BELUM ada

Backlog di `plans/next-enhancements.md` mencatat 51 task berstatus TODO berbanding 5 yang DONE. Berikut yang paling perlu diketahui, dikelompokkan menurut dampaknya.

7a. Risiko keamanan & data

BELUM ADA	KENAPA PENTING
Ganti password yang bertahan	Semua akun lahir dengan password sama, yaitu kata "password". Lebih parah, seeding di <code>db/init.ts</code> memakai <code>ON CONFLICT DO UPDATE SET password</code> dan jalan tiap container start — jadi setiap restart backend, semua password kembali ke "password". Belum ada layar ganti password sama sekali.
JWT_SECRET wajib diisi	Kalau backend/.env tidak mengisinya, kode jatuh ke nilai cadangan yang tertulis di source. Siapa pun yang tahu nilai itu bisa membuat token palsu untuk toko mana pun.
Enkripsi jalur LAN	Alamat LAN masih <code>http://</code> polos — token bearer dan seluruh isi DO lewat WiFi toko tanpa enkripsi. Jalur ngrok sudah HTTPS, jalur LAN belum.
Pembatasan log sensitif	Dio mencatat seluruh isi request/response — termasuk token dan data pribadi (nama penerima, koordinat GPS, isi DO) — dan ini tidak dimatikan di build release, jadi bocor ke adb logcat di lapangan.
Penandatanganan APK	APK release masih pakai kunci debug. Cukup untuk instalasi internal, belum layak Play Store.
Rencana backup	Volume <code>paddleocr_pgdata</code> dan folder <code>backend/uploads/</code> belum punya prosedur backup/restore terdokumentasi. Ini risiko terbesar di daftar ini.

7b. Kebenaran data

BELUM ADA	KENAPA PENTING
Penegakan cek blur	Cek blur sudah ada tapi masih kosmetik — tombol "Unggah Dokumen" tidak dikunci saat foto ditandai blur, jadi foto buram tetap bisa dikirim.
SKU dari server	Editor DO memvalidasi SKU terhadap daftar bawaan yang di-compile ke dalam aplikasi (<code>lib/data/master_sku.dart</code>), bukan endpoint <code>/api/v1/master/skus</code> yang hidup. SKU baru tidak lolos validasi sampai APK dibangun dan disebar ulang.
Penguncian simpan	Dokumen dengan SKU tak terdaftar, atau dengan nol item, masih bisa disimpan dan dicetak.
Pembersihan field dummy	Alur Product Scan masih mengirim nilai hardcoded sebagai data asli — noSo '1002003004' dan platTruk 'B 1234 PFM' — dan itu tersimpan di server seolah nyata. Ada pula fallback nama toko ke satu nama toko asli yang bisa salah-atribusi data.
Guard perubahan belum tersimpan	Tekan tombol back di editor langsung membuang koreksi manual tanpa peringatan. Hapus dokumen pending juga tanpa dialog konfirmasi.

7c. Pengalaman pemakaian di lapangan

BELUM ADA	KENAPA PENTING
Failover di tengah sesi	Alamat server hanya di-resolve sekali saat aplikasi dibuka. Kalau petugas berpindah dari WiFi ke seluler, aplikasi tidak pindah endpoint sendiri — harus tutup-buka aplikasi.
Alamat bisa diubah tanpa build ulang	Kedua URL masih konstanta di kode. Tiap IP server berubah berarti build dan sebar APK ulang ke semua toko.
Umpan balik GPS gagal	Semua jalur kegagalan lokasi hanya mencatat log lalu diam — petugas tidak diberi tahu, dokumen tetap terkirim tanpa geotag.
Penanda data basi	Kalau ambil data gagal, daftar dokumen tetap tampil seperti biasa tanpa tanda bahwa yang ditampilkan data cache lama.
Pesan error yang jujur	Kegagalan jaringan dilaporkan sebagai "Gagal mengekstrak data (Timeout)" — masalah WiFi tersamar jadi masalah OCR, menyulitkan diagnosis.
Pembersihan file foto	Foto yang sudah tersinkron tidak pernah dihapus dari penyimpanan HP — hanya catatan Hive-nya yang hilang. Penyimpanan tumbuh tanpa batas.
Badge antrean di layar kamera	Petugas yang meninggalkan layar dokumen tidak bisa melihat masih ada upload berjalan atau macet.

7d. Fitur yang direncanakan tapi belum dibangun sama sekali

BELUM ADA	KETERANGAN
Menu Stok	Seluruh alur DO ke stok (\$9 backlog): model data stok, layar input stok, menu daftar/detail stok, dan penulisan ulang Product Scan agar memakai batch stok sungguhan. Belum ada satu pun.
Jalur cepat kasir	Konfirmasi kedaluwarsa satu-ketuk di kasir, badge inferred, dan mode burst capture untuk kamera terpasang (\$10 backlog). Masih desain.
Switch Demo/Live	Mode data palsu untuk demo belum ada. Catatan: docker-compose.demo.yml memakai kata "demo" untuk hal berbeda — itu mode produksi, bukan mode data palsu. Jangan tertukar.
Switch Cloud/Local	Pilihan deployment cloud vs on-premise belum ada.

7e. Belum ada di luar kode

- **Runbook “kalau rusak, lakukan ini”** — gejala, penyebab tersering, perintah pengecekan. Misalnya HP tidak bisa login, OCR menggantung, GPU kehabisan memori, container tidak mau start.
- **Spesifikasi & akses server** — GPU apa, RAM berapa, OS-nya, siapa pegang akses SSH, dan apa yang terjadi kalau server mati lampu. Container sudah restart: unless-stopped, tapi Docker daemon-nya sendiri belum dipastikan.
- **Prosedur distribusi APK** — siapa yang build, dibagikan lewat apa, dan bagaimana memberi tahu petugas saat alamat server berubah.

- **Screenshot tiap layar** — folder screenshots/ sudah berisi tangkapan layar nyata dan dipakai oleh dokumen handover besar; ringkasan ini sengaja tidak mengulanginya.

8. Aturan kerja yang wajib diikuti penerus

ATURAN	ISINYA
Data rahasia	backend/sources/ dan backend/uploads/ berisi data klien asli — master SKU, vendor, customer, foto DO nyata, label ground truth. Jangan di-export, di-log, atau dikirim ke layanan luar.
Dua set aturan terpisah	Aturan di akar repo (CLAUDE.md + AGENTS.md) hanya mengatur aplikasi Flutter. Untuk apa pun di dalam backend/, ikuti backend/CLAUDE.md dan backend/AGENTS.md. Jangan dicampur.
Jangan ubah parser diam-diam	Pipeline OCR punya benchmark akurasi beku. Setiap perubahan parser harus dijalankan lewat harness regresi akurasi (perintahnya di backend/CLAUDE.md).
Manfaatkan knowledge graph	Ada graph struktur kode di graphify-out/ yang dibangun otomatis lewat git hook. Untuk pertanyaan “apa yang memanggil X” atau “apa yang rusak kalau Y diganti nama”, tanya ke situ dulu.
Batas 256 baris	File kode yang melewati 256 baris harus dipecah saat disentuh. Berlaku untuk aplikasi Flutter; backend/ punya aturannya sendiri.

Sumber: docker-compose.yml · docker-compose.demo.yml · backend/nginx.conf · backend/.env.example · lib/config/app_config.dart · backend/db/init.ts · backend/db/migrations/ · src/app/api/v1/auth/login/route.ts · src/app/api/v1/documents/upload/route.ts · plans/next-enhancements.md · docs/feature-list.md