AlgoDatta — Spec-1 Gap Analysis & Implementation Plan

Date: 2025-08-10

- \*\*Spec:\*\* `Spec-1-auto Trading Platform (1).pdf`

- \*\*Goal:\*\* Identify all gaps vs spec and outline a practical, phased implementation plan that turns the current codebase into a working DhanHQ + TradingView automated trading platform (paper & live).

---

## 2) Executive Summary of Missing / Incomplete Items

\*\*Frontend (Next.js)\*\*

- [ ] Login/auth UI & session handling

- [ ] Strategy Editor (incl. Pine message preview), Deploy-to-Webhook UX

- [ ] Realtime Execution Status view (stream/log console)

- [ ] Reports dashboard with filters & CSV export

- [ ] Admin Panel (user list, disable user, broker reset)

- [ ] Page routing hygiene (App Router only; remove `pages/\*` conflicts; Tailwind configured)

\*\*Backend (FastAPI)\*\*

- [ ] Auth service (JWT, register/login/me, password hashing)

- [ ] Strategy Manager (CRUD, toggle, deploy, webhook secret issuance)

- [ ] TradingView Webhook listener (shared secret, schema validation, idempotency key)

- [ ] Trade Executor service (route to Paper or Dhan live broker adapter)

- [ ] Paper Trade engine (fills, positions, PnL, fees/slippage model)

- [ ] Broker adapter (DhanHQ: connect, profile, holdings, positions, place/cancel, order status)

- [ ] Alerts/Notifications (Telegram + Email), message templates

- [ ] Logging & Retry (dead-letter/retry queue for webhook/executions)

- [ ] Reports service (summary metrics + CSV streaming endpoint)

\*\*Database / Models\*\*

- [ ] `users`, `strategies`, `alerts`, `executions`, `paper\_trades` tables

- [ ] `notifications` fully used end-to-end; `brokers` relations finalized

- [ ] Alembic migrations & seeds (admin user, sample strategy)

\*\*DevOps / Tooling\*\*

- [ ] Docker Compose (frontend + backend + Postgres + pgAdmin)

- [ ] Jenkins/CI pipeline (build/test/migrate/deploy)

- [ ] Pytest suite, Postman collection

- [ ] `.env` management for backend & frontend

---

## 3) Detailed Gap List (By Layer)

### 3.1 Frontend (Next.js App Router)

- \*\*Authentication\*\*

- Missing login/register pages, token storage/refresh, protected routes guard.

- \*\*Strategy Manager UI\*\*

- Create strategy (name, symbol, timeframe, qty, mode: paper/live)

- Edit/clone/delete; toggle enable/disable

- “Deploy” shows the webhook URL + secret, and a copyable TradingView alert JSON

- \*\*Executions View\*\*

- Live stream (server-sent events or polling) of alerts → orders → fills

- Per-execution detail drawer with request/response JSON and logs

- \*\*Reports\*\*

- Filters: date range, strategy, symbol, mode

- KPIs: Win%, Avg PnL, Sharpe (simple), Max DD, Total Trades

- Export CSV button (front-end calls `/api/reports/export?…`)

- \*\*Admin Panel\*\*

- User list, role badges, enable/disable, reset broker token

- \*\*Hygiene\*\*

- Remove legacy `pages/\*`; use only `app/\*`

- Tailwind configured and loaded once

- Centralized API base via `NEXT\_PUBLIC\_API\_BASE`

### 3.2 Backend (FastAPI)

- \*\*Auth Service\*\*

- `/api/auth/register`, `/login`, `/me`, refresh/rotate tokens; password hashing (bcrypt), JWT (HS256), RBAC (admin, user)

- \*\*Strategy Manager\*\*

- `/api/strategies` CRUD, `/toggle`, `/deploy` (issues `webhook\_secret`)

- \*\*TradingView Webhook\*\*

- `/api/webhook/tradingview` (POST): validates shared secret, schema, idempotency key; persists an `alert` row; enqueues execution

- \*\*Execution Pipeline\*\*

- `TradeExecutor`: normalizes order request, routes to `PaperEngine` or `DhanBroker`

- Idempotent submission, retry policy (exponential backoff), DLQ table for failed actions

- \*\*Broker Adapter: DhanHQ\*\*

- Connect (OAuth/token flow), profile/holdings/positions, place/cancel order, order status polling

- \*\*Paper Engine\*\*

- Naive matching engine (market/limit), positions & PnL, fees/slippage model, timestamps in IST

- \*\*Notifications\*\*

- Telegram bot & SendGrid email; opt-in per user/strategy; message templates (alert received, order placed, order filled, error)

- \*\*Reports\*\*

- Summary aggregates; time-bucketed stats; CSV streaming endpoint

- \*\*Logging\*\*

- Structured logs (JSON), correlation IDs per alert/execution, audit trail

### 3.3 Database / Entities (proposed minimal schema)

- \*\*users\*\*: id, email, password\_hash, role, status, created\_at

- \*\*brokers\*\*: id, user\_id FK, name, access\_token, refresh\_token, status, connected\_at, updated\_at

- \*\*strategies\*\*: id, user\_id FK, name, symbol, timeframe, qty, mode(paper|live), status, webhook\_secret, created\_at, updated\_at

- \*\*alerts\*\*: id, strategy\_id FK, idempotency\_key, payload\_json, received\_at, processed\_at, status, error\_text

- \*\*executions\*\*: id, strategy\_id FK, side, qty, price, mode, broker\_order\_id, status, pnl, created\_at, updated\_at

- \*\*paper\_trades\*\*: id, strategy\_id FK, side, qty, price, fill\_ts, position\_qty, avg\_price

- \*\*notifications\*\*: id, user\_id FK, type(telegram|email), destination, verified, created\_at

- \*\*error\_logs\*\*: id, user\_id FK (nullable), context, message, stack, created\_at

---

## 4) Phased Implementation Plan

\*\*Phase 0 — Project Hygiene (1st)\*\*

- Remove `pages/\*` conflicts; consolidate to `app/\*`

- Tailwind setup; shared layout; API base via env

- Central error boundary & toast system

\*\*Phase 1 — Data Layer\*\*

- SQLAlchemy models + Alembic migrations

- Seed admin user; seed sample strategy

\*\*Phase 2 — Auth & Admin\*\*

- JWT auth, password hashing

- Admin endpoints & UI (user list, disable/enable, reset broker token)

\*\*Phase 3 — Strategies & Webhook\*\*

- Strategy CRUD/toggle/deploy

- TradingView webhook: validation, persistence, idempotency

\*\*Phase 4 — Execution Pipeline\*\*

- TradeExecutor + PaperEngine (PnL & positions)

- Dhan broker adapter (profile/positions/place/cancel/status)

- Retry & DLQ tables, correlation IDs

\*\*Phase 5 — Reports\*\*

- Aggregated metrics + CSV export endpoint

- Reports UI with filters + export

\*\*Phase 6 — Notifications\*\*

- Telegram + Email integration, templates, opt-ins

\*\*Phase 7 — Frontend Integration\*\*

- Broker page (connect, profile, holdings, positions)

- Strategies page (table, toggles, deploy modal)

- Executions page (live log stream, details)

- Reports page (KPIs, table, CSV)

- Admin page (users, broker reset)

\*\*Phase 8 — CI/CD & Ops\*\*

- Docker Compose (frontend, backend, postgres, pgadmin)

- Jenkins: build, test, migrate, deploy

- Env hardening, secrets handling

\*\*Phase 9 — Tests & Tooling\*\*

- Pytest unit & API tests

- Postman collection, sample envs

- Load-testing webhook endpoint (idempotency check)

---

## 5) Key Files to Add/Update (Illustrative)

backend/

app/

main.py

core/

config.py

security.py

logging.py

models/

user.py

broker.py

strategy.py

alert.py

execution.py

paper\_trade.py

notification.py

error\_log.py

\_\_init\_\_.py

schemas/

auth.py

strategy.py

execution.py

broker.py

notification.py

routes/

auth.py

strategies.py

webhook.py

executions.py

reports.py

broker\_dhan.py

admin.py

notifications.py

services/

trade\_executor.py

paper\_engine.py

broker\_dhan.py

notifier.py

retry\_queue.py

db/

session.py

base.py

migrations/ (alembic)

frontend/

app/

layout.tsx

page.tsx

broker/page.tsx

strategies/page.tsx

executions/page.tsx

reports/page.tsx

admin/page.tsx

lib/api.ts

components/

StrategyTable.tsx

StrategyForm.tsx

DeployModal.tsx

LiveLogConsole.tsx

KpiCards.tsx

CsvExportButton.tsx

---

## 6) Environment Variables (.env examples)

\*\*Backend\*\*

SECRET\_KEY=changeme

ACCESS\_TOKEN\_EXPIRE\_MINUTES=60

DATABASE\_URL=postgresql+psycopg2://postgres:postgres@db:5432/algodatta

DHAN\_API\_KEY=your\_key

DHAN\_API\_SECRET=your\_secret

TELEGRAM\_BOT\_TOKEN=your\_bot\_token

SENDGRID\_API\_KEY=your\_sendgrid\_key

TZ=Asia/Kolkata

\*\*Frontend\*\*

NEXT\_PUBLIC\_API\_BASE=http://localhost:8000

---

## 7) API Contracts (Samples)

\*\*TradingView Alert → Webhook JSON\*\*

{

"secret": "{{webhook\_secret}}",

"idempotency\_key": "{{strategy\_id}}-{{timestamp}}-{{bar}}",

"symbol": "NATURALGAS",

"side": "BUY",

"qty": 1,

"order\_type": "MARKET",

"price": null,

"timeframe": "5m",

"extras": {

"source": "TradingView",

"note": "Breakout entry"

}

}

\*\*CSV Export\*\*

- `GET /api/reports/export?from=2025-01-01&to=2025-08-10&strategy\_id=...`

- Response: `text/csv` streamed

---

## 8) Acceptance Criteria

- Authenticated user can create & deploy a strategy; TradingView alert triggers an execution

- Paper mode produces fills and positions with PnL; live mode routes to DhanHQ and returns broker order id

- Executions and alerts auditable with logs and correlation ID

- Reports page shows KPIs and allows CSV export

- Notifications reliably sent for fill/error events

- CI pipeline builds, tests, migrates DB, and deploys containers successfully

---

## 9) Next Steps

1. Approve this plan and I’ll start with \*\*Phase 0–2\*\*.

2. Share your preferred Telegram/email creds for a non-production sandbox.

3. Confirm whether to keep both \*\*paper\*\* and \*\*live\*\* behind a per-strategy toggle by default.

---

\*Prepared for Kotra (IST)\*