**Reasoning & Requirements Analysis**

* Purpose
  + Build an agentic AI trading system that:
    - Scans crypto spot coins and F&O (futures and options) on CoinDCX for short-term/swing opportunities.
    - Scores opportunities with confidence, potential return, target price, and stop loss.
    - Auto-invests with percentage-wise fund sharing when confidence exceeds a configurable threshold.
    - Monitors open positions continuously and exits on TP/SL or risk triggers.
    - Presents a multi-page Streamlit UI with real-time data, interactive charts, and full portfolio management.
* Target users
  + Active crypto traders familiar with short-term trading, comfortable with automated execution.
  + Quant-inclined users who want an explainable AI agent to assist with idea generation and execution.
  + Users with CoinDCX accounts and API keys (spot and futures).
* Trading scope, assumptions, and definitions
  + Instruments: CoinDCX Spot symbols and Futures (F&O). Options are limited on many Indian exchanges; we treat “F&O” primarily as futures on CoinDCX; if/when options endpoints are available, we plug into the same adapter pattern.
  + Timeframes:
    - Short-term: intraday to 24 hours
    - Swing: 2–10 days
    - Configurable in Strategy Settings.
  + Entry orders: Market or limit (user configurable). Prefer limit for futures to control slippage; market allowed with max slippage guard.
  + Targets/Stops: By default ATR- and support/resistance-based, risk-reward target 1.5–2.5x.
  + Confidence: Ensemble score from technical factors + regime filters + optional LLM synthesis via CrewAI.
* Key constraints and risks
  + Exchange API limits and reliability: Must throttle, retry, and fail gracefully.
  + Price streams: Use CoinDCX websockets for real-time; fall back to polling.
  + Partial fills, latency, slippage: Execution engine must reconcile and adjust.
  + Futures risk: Leverage, liquidation; hard risk controls required (max leverage, min maintenance margin checks).
  + No paper trading in CoinDCX: Provide built-in simulation mode.
  + Security: API keys must be encrypted and never logged; include kill-switch and “Manual-only” mode.
  + Compliance: This is tooling, not financial advice; the user controls risk parameters.
* Success criteria
  + End-to-end from signal to executed orders, visible in dashboard.
  + Configurable confidence threshold, risk/position sizing, and fund distribution.
  + Real-time monitoring with accurate PnL, TP/SL enforcement, and alerts.
  + Explainability: Show why an instrument was chosen, including the factors and CrewAI rationale.
* Open questions (proceeding with defaults but keep configurable)
  + Target holding count per strategy (default 5).
  + Allocation between spot and futures (default 60% spot, 40% futures).
  + Max leverage and per-position risk (default max leverage 3x, risk per position 0.5–1.0% of total equity).
  + Entry order type defaults (spot: market with slippage guard; futures: post-only limit).
  + Confidence threshold (default 0.65).
  + Supported markets universe (top 150 by market cap + liquidity filter).
  + Options support availability on CoinDCX (assume futures first; add options when API access confirmed).
* Technical considerations
  + Python-centric stack: Streamlit UI, CrewAI for multi-agent reasoning, data adapters for CoinDCX REST/WS.
  + Asynchronous tasks for streaming, scheduling, and monitoring using asyncio + APScheduler.
  + Persistence with SQLite for simplicity (upgradeable to Postgres).
  + Backtesting and indicators with vectorbt/pandas-ta/ta-lib; ensure deterministic numeric outputs that AI agents can reason about.
  + Containerization for reproducibility; secrets via environment or OS keyring.

**Feature Breakdown and UX/UI Design**

* Multi-page Streamlit UI
  + Dashboard (Live)
    - Account balances (spot/futures), equity curve, exposure heatmap.
    - Open positions with TP/SL/Trailing SL, unrealized PnL, margin utilization.
    - Kill-switch toggle, Auto-Trade toggle, current confidence threshold slider.
    - Real-time tickers and mini-charts (candlestick with volume).
  + Screener & Signals
    - Universe filter, liquidity filter, timeframe selection.
    - Table of candidates with: symbol, timeframe, confidence score, expected return (%), entry, TP, SL, R:R, rationale.
    - Interactive charts (candlestick, indicators: EMA, RSI, MACD, ATR, OI/funding if available).
    - Manual override: exclude/include, edit targets/stops, queue for execution.
  + Strategy Config
    - Strategies (Momentum, Breakout, Mean-reversion, Volatility breakout, Funding/OI bias).
    - Confidence model weights & threshold.
    - Position sizing: fixed fraction, volatility-adjusted, risk per trade, max positions.
    - Allocation: % Spot vs % Futures; Fund sharing logic (equal weight, risk parity, confidence-weighted).
    - Entry/Exit rules, trailing SL, cool-down windows, no-trade times.
  + Portfolio & Orders
    - Open orders, order history, fill details, commissions, slippage.
    - Positions with performance metrics, pyramiding/scale-out controls.
    - Rebalance and close-all buttons.
  + Backtesting & Logs
    - Run quick backtests on select universes/timeframes.
    - Performance metrics (CAGR, Sharpe, MaxDD, Win rate, Expectancy).
    - System logs, CrewAI reasoning snapshots, audit trail of decisions.
  + API & Keys
    - Secure onboarding for API keys (spot/futures).
    - Connectivity tests, permissions check.
    - Mode: Live, Paper/Simulation, Dry-run.
  + Alerts
    - Email/Telegram/Discord webhook integration.
    - Alert rules: entry triggered, TP/SL hit, max drawdown, API failures.
  + Risk Center
    - Global constraints: max daily loss, max leverage, per-asset cap, correlation cap.
    - Circuit breakers: volatility spikes, exchange maintenance detection.
    - Compliance settings and disclaimers.
* Core trading features
  + Opportunity detection
    - Technical factor engine computes features and scores per instrument/timeframe.
    - CrewAI agents synthesize features into a structured JSON recommendation with confidence, expected return, TP, SL, rationale.
  + Auto allocation and execution
    - When confidence >= threshold, allocate funds by configured scheme.
    - Place orders (spot/futures) respecting risk constraints and allowed leverage.
    - Use bracket (OCO) if supported; else monitor TP/SL via price watcher.
  + Continuous monitoring
    - Websocket price streams, periodic polling fallback.
    - Exit on TP/SL, trailing SL, time-stop, or global risk triggers.
    - Handle partial fills, reprice/edit orders, cancel stale orders.
  + Explainability
    - Show the features contributing to the score and CrewAI rationale text with links to charts/timeframes.
* UX principles
  + Safety first: always visible kill switch, clear mode indicator (Live vs Paper).
  + Transparency: show raw numbers, explain rationales, and provide logs.
  + Control: user can override auto decisions and edit orders before execution.
  + Responsiveness: real-time updates without blocking; clean state management.
  + Accessibility: dark/light themes, mobile-friendly Streamlit layouts.

**Technology Stack Justification**

* Python
  + Primary language for data analysis, ML/TA libraries, and exchange API clients.
* Streamlit
  + Rapid development of multi-page data apps; easy deployment.
  + Real-time-ish updates via session\_state, st.cache\_data, st.cache\_resource; supports Plotly/Altair charts.
* CrewAI
  + Agentic orchestration to combine numeric factor outputs with explainable reasoning and decision validation.
  + Multi-agent flow: Research, Portfolio, Risk, Execution, Monitor, Critic.
* CoinDCX API
  + REST for account, orders, and historical candles.
  + Websockets for tickers/orderbook and possibly user execution updates.
* Supporting libraries
  + Asyncio, websockets, httpx/requests for networking.
  + APScheduler for periodic jobs.
  + pandas, numpy, pandas-ta/ta-lib, vectorbt for indicators/backtests.
  + Pydantic for configs and typed models; SQLAlchemy + SQLite for persistence.
  + Plotly for interactive charts; streamlit-aggrid for rich tables.
  + cryptography/keyring for secure API key storage.
  + Tenacity for robust retries; loguru for structured logging.

Rationale: This stack stays within Python-centred tooling, integrates CrewAI for agent workflows, uses Streamlit for rapid UI, and speaks directly to CoinDCX APIs. It scales vertically for a single user and can be modularized later.

**Architecture Overview and Data Model Sketches**

* High-level components
  + UI (Streamlit multipage)
    - Renders data from Services via an in-process service layer.
    - Publishes user actions (toggles, edits, executes) to Services.
  + Services (in the same process or as modules)
    - MarketDataService: REST/WS to CoinDCX, caching candles, ticks.
    - SignalService: Feature computation, strategy scores.
    - CrewAI Orchestrator: Agents consume structured features, produce JSON recommendations + rationale.
    - PortfolioService: Positions, orders, allocation, PnL, risk metrics.
    - ExecutionService: Order routing to CoinDCX Spot/Futures adapters with risk guards.
    - MonitorService: TP/SL checks, trailing SL, time-stop, global risk triggers.
    - Scheduler: APScheduler jobs for scanning and monitoring.
    - Persistence: SQLAlchemy ORM to SQLite; JSON logs of agent outputs.
    - Alerts: Email/Telegram/Discord integrations.
  + Exchange Adapters
    - SpotAdapter and FuturesAdapter implement a common interface for symbols, balances, place/cancel orders, positions, leverage.
    - Websocket client for market streams; user stream if available.
  + Config & Secrets
    - Pydantic settings; environment variables for API keys; optional OS keyring; AES-encrypted local file.
* CrewAI agents and tools
  + MarketDataAgent: ensures clean datasets; validates symbol mapping; fetches candles.
  + TechnicalSignalAgent: computes indicators and forms structured features.
  + ResearchAgent (LLM): consumes features, market regime context; outputs per-asset JSON: {confidence, expected\_return\_pct, entry, tp, sl, rationale}.
  + RiskAgent: applies risk overlays (volatility, correlation, exposure, leverage).
  + PortfolioAgent: allocates capital across shortlisted assets (confidence-weighted or risk parity).
  + ExecutionAgent: final sanity checks, builds order basket, calls ExecutionService.
  + MonitorAgent: watches positions for TP/SL and anomalies; suggests adjust/exit.
  + CriticAgent: validates numeric consistency; flags outliers or hallucinations.
* Data models (sketch)
  + Asset
    - id, symbol, type (spot/futures), base, quote, tick\_size, lot\_size, min\_notional, leverage\_limits
  + Candle
    - id, symbol, timeframe, ts\_open, open, high, low, close, volume, oi, funding\_rate
  + FeatureSnapshot
    - id, symbol, timeframe, ts, features: JSON (RSI, MACD, ATR, EMAs, breakouts, vol, trend, funding, OI)
  + Signal
    - id, symbol, timeframe, ts, strategy, confidence, expected\_return\_pct, entry, tp, sl, rationale, status (new/approved/ordered)
  + StrategyConfig
    - id, name, params JSON (weights, thresholds, stop/target rules, max\_positions, allowed\_markets, leverage caps)
  + Order
    - id, exchange\_order\_id, symbol, side, type, qty, price, status, ts\_created, ts\_updated, sl\_price, tp\_price, parent\_order\_id, reduce\_only, time\_in\_force
  + Position
    - id, symbol, side, entry\_price, qty, realized\_pnl, unrealized\_pnl, leverage, sl, tp, trailing\_params, ts\_open, ts\_close, status
  + PortfolioSnapshot
    - id, ts, equity, cash\_spot, cash\_futures, margin\_used, exposure by asset class JSON
  + Alert
    - id, ts, level, message, context JSON
  + ApiCredentials
    - id, label, key\_id, encrypted\_secret, type (spot/futures), permissions JSON
* Core workflows
  + Scan → Score → Allocate → Execute → Monitor
    - Scheduler triggers scan (e.g., every 5 minutes).
    - MarketDataService pulls latest candles; Feature engine computes features.
    - CrewAI ResearchAgent consumes features + regime; outputs candidates with confidence/TP/SL.
    - RiskAgent filters by thresholds; PortfolioAgent computes allocation weights based on configured fund sharing.
    - ExecutionAgent submits orders via adapters with guards; link TP/SL as bracket or via MonitorService.
    - MonitorService runs continuously; exits on TP/SL or risk; updates UI and logs; Alerts if events occur.
* Percent-wise fund sharing
  + Global allocation: e.g., 60% Spot, 40% Futures.
  + Within each bucket: weight by normalized (confidence × expected\_return\_pct / volatility).
  + Enforce per-position risk cap and min notional.
  + Respect confidence threshold: only allocate to assets above threshold.
* Risk controls
  + Pre-trade:
    - Check leverage availability, margin, min notional, symbol tradability.
    - Enforce per-position risk, correlation cap (avoid overexposure to highly correlated assets).
  + Post-trade:
    - Attach SL/TP if supported; else register watchers.
    - Trailing SL using ATR multiple or % trail.
  + Global:
    - Max daily loss; if hit, auto-disable Auto-Trade and notify.
    - Kill switch button; disable all new orders and cancel open ones if configured.
* Error handling and resilience
  + Retries with backoff for API calls.
  + Idempotent order placement using clientOrderId.
  + Reconciliation loop to match local records with exchange state.
  + Graceful degradation if websockets fail (polling fallback).
  + Extensive logging and audit trails.
* Example file structure
  + app/
    - streamlit\_app.py (root multipage index)
    - pages/
      * 1\_Dashboard.py
      * 2\_Screener\_and\_Signals.py
      * 3\_Strategy\_Config.py
      * 4\_Portfolio\_and\_Orders.py
      * 5\_Backtesting\_and\_Logs.py
      * 6\_API\_and\_Keys.py
      * 7\_Risk\_Center.py
    - services/
      * market\_data.py
      * signals.py
      * portfolio.py
      * execution.py
      * monitor.py
      * scheduler.py
      * adapters/
        + coindcx\_spot.py
        + coindcx\_futures.py
    - ai/
      * crew.py
      * agents.py
      * tools.py
      * prompts/
    - models/
      * db.py
      * orm.py
      * schemas.py
    - utils/
      * config.py
      * logging.py
      * security.py
      * charts.py
    - tests/
* Pseudocode snippets (condensed, illustrative)
  + CrewAI agents wiring
    - Define a ResearchAgent that takes numeric features and produces structured JSON. Then pass through a CriticAgent to validate outputs. Finally, Risk/Portfolio/Execution agents convert to orders.
  + Signal to order conversion logic
    - Normalize weights using confidence × expected\_return / volatility, cap per-position risk, compute qty based on entry price and leverage, place orders via adapters, attach SL/TP or register monitors.
  + Monitor loop
    - Subscribe to price streams; update unrealized PnL; if price >= TP or <= SL (by side), close position; update DB; alert.

Note: Real CoinDCX endpoints and payloads must be confirmed from current API docs; adapt fields and websockets accordingly.

**Implementation Roadmap (phases, milestones)**

* Phase 0: Foundations (Week 1)
  + Setup repo, env, dependency management.
  + Build config management (Pydantic), secure key storage, logging.
  + Implement ORM models and DB migrations (SQLite).
  + Create CoinDCX adapters (Spot + Futures) with connectivity tests, rate limiters, retry/backoff.
* Phase 1: Data & Signals (Weeks 2–3)
  + MarketDataService for REST candles + WS tickers; caching layer.
  + Feature engine with pandas-ta/ta-lib; compute RSI, MACD, EMAs, ATR, volatility, breakouts, trend filters, funding/OI if available.
  + Build baseline scoring (numeric ensemble) independent of LLM to ensure robust outputs.
  + Streamlit pages: Dashboard skeleton, Screener with table, basic charts.
* Phase 2: CrewAI Integration (Week 4)
  + Define agents (Research, Risk, Portfolio, Execution, Monitor, Critic).
  + Tools to feed structured features; produce JSON recommendations with confidence, expected return, TP, SL, rationale.
  + Display rationales and scores in UI; allow manual approves.
* Phase 3: Execution & Monitoring (Weeks 5–6)
  + ExecutionService with order placement, clientOrderId, bracket orders or simulated TP/SL.
  + MonitorService with WS/polling, TP/SL checks, trailing SL, time-stop.
  + Portfolio & Orders page: live PnL, positions table, order history.
  + Auto-Trade toggle; fund allocation modes; confidence threshold control.
* Phase 4: Risk Center & Alerts (Week 7)
  + Implement global risk constraints, kill switch, daily loss limits.
  + Alerts integrations (Telegram/Email/Webhooks).
  + Circuit breaker logic for volatility spikes and API outages.
* Phase 5: Backtesting & Refinement (Weeks 8–9)
  + Quick backtester for numeric strategies; parameter sweeps.
  + Performance reports; store backtests.
  + Improve scoring calibration using historical hit ratios; calibrate confidence mapping.
* Phase 6: Hardening & Deployment (Week 10)
  + Unit/integration tests; paper/simulation burn-in.
  + Optimize WS handling and UI responsiveness.
  + Containerize; deployment guide; runbooks.
  + Security review; secrets handling.

Deliverables each phase: code, tests, docs, and demo videos. Gate to live trading only after paper trading success and risk sign-offs.

**Final Summary & Recommendations**

* This design delivers an agentic AI trading system using Python, CrewAI, Streamlit, and CoinDCX APIs to identify and trade short-term/swing opportunities across spot and futures. It combines a deterministic numeric factor engine with CrewAI’s explainable reasoning, producing confidence, expected return, target, and stop for each candidate. When confidence exceeds a user-set threshold, the app allocates funds by configured percentages and executes orders, with continuous TP/SL monitoring and robust risk controls.
* The UI is a multi-page Streamlit app with a real-time dashboard, screener with interactive charts, portfolio/order management, strategy configuration, backtesting, and risk/alert centers—giving traders full control and transparency.
* Start with a conservative, numeric-first signal pipeline and add CrewAI for explainability and incremental alpha. Use paper trading to calibrate confidence thresholds, risk sizing, and targets/stops before going live. Maintain strict security for API keys, include a global kill switch, and implement strong monitoring and logging.
* Next steps: confirm CoinDCX futures endpoints and OCO support; finalize the trading universe; choose default thresholds; begin Phase 0–1 implementation and iterate with paper trading feedback.