Technical Specification: FDA-Linked Options Trading Dashboard (Streamlit Desktop App)

1. Architecture Overview

This is a Python-based desktop application using Streamlit for the frontend dashboard.

It integrates PostgreSQL for persistent data storage, and APIs like Alpaca for trading data and Twilio for notifications.

Background jobs such as scraping or alerts are handled by Celery and Redis.

2. Technology Stack

- Frontend/UI: Streamlit (runs in localhost browser)
- Visualization: Plotly, Altair, or Matplotlib
- Backend/Logic: Pure Python scripts or FastAPI (optional)
- Database: PostgreSQL
- Background Jobs: Celery + Redis (for scrapers, alerts)
- Notifications: Twilio (SMS), SendGrid (Email)
- Scraping: BeautifulSoup, Playwright

3. Data Models

a. fda_events:

- id, drug_name, company_name, ticker, event_type, event_date, source_url, sentiment, status, created_at

b. options_positions:

- id, alpaca_order_id, ticker, strategy_type, entry_price, current_price, quantity, entry_date, status, linked_event_id, break_even_low, break_even_high, iv_entry, iv_current

c. trade_history: Similar to options_positions but closed trades only

d. users (optional): id, email, api_keys (JSON)

4. Key External Integrations

- Alpaca API: trading and market data
- Market Data: Polygon, Tradier (for options chain and IV)
- Web Scrapers: FDA, EMA, ClinicalTrials.gov
- Notifications: Twilio for SMS, SendGrid for Email

5. Frontend Features (Streamlit)

- Active Positions Table with real-time P&L, IV, and links to catalyst
- Break-even chart (Plotly)
- Upcoming FDA/EMA Events Table
- Trade History Log with filters
- Event Management: raw data viewer, manual linking
- Optional Alert Config: SMS/email toggle, thresholds

6. Deployment

- Local CLI usage: `streamlit run dashboard.py`
- Optionally bundle into a desktop app using PyInstaller or Eel
- PostgreSQL runs locally or in a Docker container
- Use .env for API keys and credentials