■ NEXUS TRADING BOT (Model-7) - SYSTEM TEARDOWN

1. System Overview

Nexus Trading Bot is a high-frequency, event-driven algorithmic trading system designed for **Binance Futures** (Crypto) and **Alpaca** (Stocks/ETFs). It utilizes a hybrid architecture combining real-time WebSocket data streams with asynchronous processing to execute trades with varying degrees of autonomy (**Watch**, **Copilot**, **Pilot**).

The core philosophy is **"Defense First, Attack Second"**, embodied by the **Sentinel Protocol** (System Shield) and high-precision strategies like **Shark Mode**.

---

2. Architecture & Tech Stack

Technology Stack

- **Language**: Python 3.10+ (Async/Await specific)
- **Framework**: `aiogram 3.x` (Telegram Interface), `asyncio` (Concurrency)
- **Exchange Clients**: `python-binance` (Async), `alpaca-trade-api` (Sync Wrapper)
- **Data Science**: `pandas`, `numpy`, `pandas_ta` (Technical Analysis), `scikit-learn` (ML Cortex)
- **Database**: PostgreSQL (Session/State), JSON (Fallback/Logs)
- **Deployment**: Design compatible with Railway/Docker.

Modular Architecture

The system is divided into clear functional zones ("Biomes"):

A. CORTEX (The Brain) - `nexus_system/cortex/`

Responsible for signal generation and strategy logic. - **Factory (`factory.py`)**: Dynamic strategy selector based on market regime. - **Strategies**: - `scalping.py`: RSI/Momentum scalper for ranging markets. - `flow.py`: Trend following (EMA/ADX). - `harpoon.py`: Reversal catcher (Bollinger Bands). - `shark.py`: **Shark Strategy** (Aggressive Shorting for crashes). - **Classifier (`classifier.py`)**: ML Model to determine market regime (Volatile/Stable).

B. SHIELD (The Defense) - `nexus_system/shield/`

Protect capital above all else. - **Risk Manager (`manager.py`)**: - **Black Swan Detection**: Monitors BTC crash (>4%) via WebSocket. - **Macro Analysis**: Polls CoinMarketCap for BTC Dominance trends. - **Circuit Breaker**: Halts trading after consecutive losses. - **Exposure Control**: Limits max position sizing.

C. UPLINK (The Senses) - `nexus_system/uplink/`

Data ingestion and normalization. - **Stream (`stream.py`)**: Manages WebSocket connections (Binance/Alpaca w/ Polygon). - **CMC Client (`cmc_client.py`)**: Polls CoinMarketCap for global metrics.

D. CORE (The Heart) - `nexus_system/core/`

Engine (`engine.py`): The central event loop.

- Listens to WebSocket ticks.
- Routes data to proper Strategy.
- Queries Risk Manager overrides.
- Dispatches signals to the Bot.

E. SERVOS (The Hands) - `servos/`

Execution and User Management. - **Trading Manager**: Handles Order Placement, SL/TP Sync, Anti-Accumulation. - **Session Manager**: Manages multi-user state (Chat IDs, API Keys). - **Personalities**: Dynamic response system (Tone/Voice).

---

3. Key Functionalities

■■ Sentinel Protocol (The Guardian)

An always-on defensive layer that overrides all other strategies. - **Black Swan Event**: If BTC crashes >4% in minutes -> Immediate **EXIT_LONG** only. Shorts remain open. - **Shark Context**: If BTC Dominance rises + Global Cap falls -> Trigger **SHARK_MODE**.

■ Shark Mode (The Predator)

- **Trigger**: "Sangria" Market (Bleeding Alts).
- **Action**: Aggressively SHORT specific targets (`SHARK_TARGETS` like SOL, MEMEs).
- **Logic**: High Leverage, Tight SL, follows downside momentum.

■ ML-Enhanced Regimes

The bot uses a Random Forest / XGBoost model to classify the market structure (Trending vs Ranging) to automatically select the best strategy: - **Scalping**: For Choppy/Sideways markets. - **Flow**: For Strong Trends.

■ Operation Modes

1. **WATCHER**: Analysis only. Sends Alerts. No trades.
2. **COPILOT**: Sends "Proposals" (Buttons). User must click [Approve].
3. **PILOT**: Full Autonomy. Executes trades instantly.

---

4. Workflows

A. The Signal Loop

1. **Tick**: WebSocket receives price update for `BTCUSDT`.
2. **Shield**: `RiskManager` checks for Black Swan.
3. **Cortex**: If safe, `StrategyFactory` selects strategy (e.g., `ScalpingStrategy`).

4. **Analysis**: Strategy calculates RSI, EMA, ADX. Returns `Signal(BUY/SELL)`.
5. **Core**: `NexusCore` dispatches Signal.
6. **Servo**: `TradingManager` executes order on Binance/Alpaca.
7. **Notify**: Telegram Message sent to user (with Personality).

B. Dashboard & Macro

- Users check `/dashboard` to see PnL and **BTC Dominance**.
- Background poller updates Macro data every 10 min from CoinMarketCap.

---

5. Directory Structure

```
Nexus-TB/ ■■■ handlers/ # Telegram Command Handlers ■■■ nexus_system/ # Core System (Brain/Shield/Uplink) ■ ■■■ core/ # Execution Engine ■ ■■■ cortex/ # Strategy Logic ■ ■■■ shield/ # Risk Management ■ ■■■ uplink/ # Data Feeds ■■■ servos/ # Utilities (DB, Trading, Auth) ■■■ system_directive.py # GLOBAL Configuration ■■■ nexus_loader.py # Main Entry Point
```

6. Credits

Engineered by **Antigravity** (Google DeepMind) & User (Fabio). Top Tier Algorithmic Trading System.