## **Project Overview**

This system will train a PPO agent to consistently achieve configurable daily profit targets and starting on futures trading data, master individual trading days through transfer learning, and deploy for live trading through the Tradovate API with robust risk controls. The solution will deliver a production-grade system with proper environment design, reward mechanisms, day mastery protocols, and seamless transition from simulation to live trading.

Project Overview The completed system will feature:

1. A production-grade Gymnasium-registered RL environment for simulating futures trading
2. Realistic market mechanics
   1. margin
   2. commission
   3. slippage
   4. OHLC/high-low P/L calculation
3. A training pipeline with transfer learning for day mastery
4. Live trading deployment via Tradovate API with comprehensive risk controls
5. Advanced visualization and monitoring capabilities
6. Containerized deployment with robust testing and documentation

Detailed Technical Approach

1. **Environment Refinement** The core TradingDataEnv (Gymnasium environment) will be optimized with:  
   * State Space: A stacked history of OHLC data plus technical indicators (EMA, SMA, Bollinger Bands, ATR, volatility, momentum) and account metrics (equity, P/L, position information).
   * Action Space: Four discrete actions (Buy Max, Sell Max, Hold, Close All)
   * Reward Function: Primarily delta in net equity with significant bonuses for hitting profit targets and severe penalties for bankruptcy
   * Termination Conditions: Bankruptcy (80% equity reserve falls below margin requirement), profit target achieved, or end of trading day

**Pseudocode sketch of the core environment structure** class TradingDataEnv(gym.Env):

def \*\*init\*\*(self, config, data\_path, day\_index=None):

self.config = config

self.data = self.\_load\_data(data\_path, day\_index)

**Action and observation spaces** self.action\_space = spaces.Discrete(4) # Buy, Sell, Hold, Close

**Observation includes market data + indicators + account state** obs\_shape = (config['history\_length'], self.\_get\_feature\_count())

self.observation\_space = spaces.Box(low=-np.inf, high=np.inf, shape=obs\_shape)

**Initialize trading state** self.reset()

def step(self, action):

**Execute action (buy/sell/hold/close)** self.\_execute\_action(action)

**Update environment state** self.\_update\_state()

**Calculate reward** reward = self.\_calculate\_reward()

**Check termination conditions** done, info = self.\_check\_termination()

**Return state, reward, done, info** return self.\_get\_observation(), reward, done, False, info

def reset(self, seed=None, options=None):

**Reset environment state (equity, position, etc.)** self.\_reset\_state()

**Return initial observation** return self.\_get\_observation(), {}

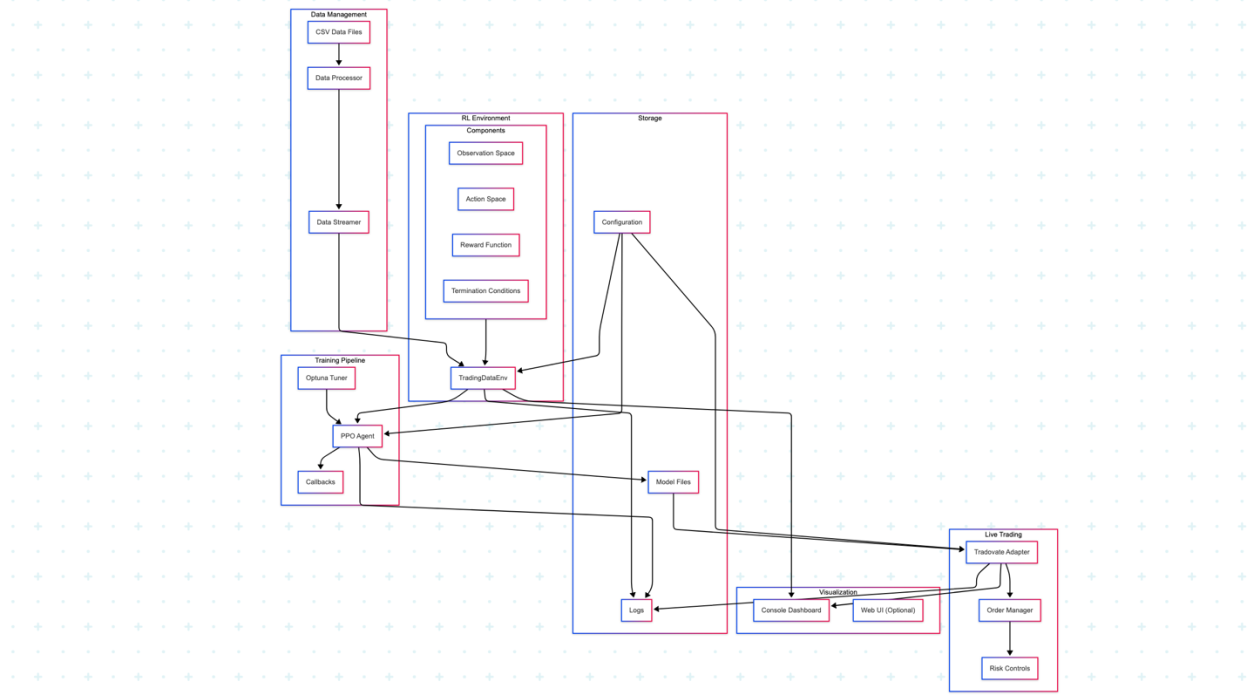
1. Data Ingestion Pipeline

Flexible CSV-to-stream simulator with the following features:

* Support for any timeframe (1m, 5m, tick)
* Conversion of timestamps to trading sessions with configurable start/end hours
* Preprocessing for technical indicators and normalization
* Random day selection for training with transfer learning

1. Day Mastery & Transfer Learning The system will implement a robust day mastery mechanism:  
   * Track success metrics per trading day (profit target hit rate, efficiency metrics)
   * Mark a day as "mastered" when:
     + At least X episodes have been run
     + ≥95% of episodes hit the profit target
     + Performance plateaus with no improvement in trade count or time over Y episodes
   * Upon mastery, increment to the next random day while preserving model weights
2. PPO Training Pipeline The training pipeline will use Stable-Baselines3 with:  
   * Hyperparameter tuning using Optuna
   * Checkpointing for model saving/resumption
   * Evaluation callbacks for tracking performance
   * Plateau detection for day mastery
   * Structured logging of all metrics
3. Live Trading Adapter The Tradovate API connector will include:  
   * Authentication and order management
   * Real-time market data streaming
   * Position tracking and risk controls
   * Emergency kill-switches
   * Comprehensive logging for compliance
4. Visualization & UI The system will provide:  
   * Fast training mode with minimal rendering
   * Human-readable console dashboard showing real-time metrics
   * Optional web/desktop UI for training control and visualization
   * Post-training analysis tools
5. Testing & Quality Assurance Comprehensive testing will include:  
   * Unit tests for environment logic, P/L calculations, margin enforcement
   * Integration tests for the training pipeline and live adapter
   * Stress testing for edge cases and error conditions
   * CI/CD pipeline for automatic testing

Here is a flow chart that describes the architecture/workflow:



1. Optimized RL Environment  
   1. Enhanced TradingDataEnv with realistic market mechanics
   2. Observation space with essential technical indicators
   3. Effective reward function for profit target achievement
2. Core Training Pipeline  
   1. Stable-Baselines3 PPO implementation for training
   2. Day mastery tracking and transfer learning
   3. Enhanced plateau detection
3. Essential Live Trading Components  
   1. Tradovate API connector with complete functionality and correctly configured commands
   2. Core risk controls and position management
   3. Advanced monitoring and logging
4. Documentation Package  
   1. Technical documentation for all components
   2. Configuration guide and examples
   3. Installation and usage instructions for both environment and live deployment

**Implementation Approach** Core Environment

Key Technical Deliverables:

* Realistic P/L tracking using high/low prices
* Proper margin enforcement with 80% equity reserve
* Commission handling on trades
* Selectable technical indicators calculated in preprocessing

**Streamlined Training Pipeline** The optimized training approach will:

* Use the PPO algorithm with targeted hyperparameters
* Implement checkpointing for model saving
* Apply transfer learning between days
* Track core metrics for day mastery

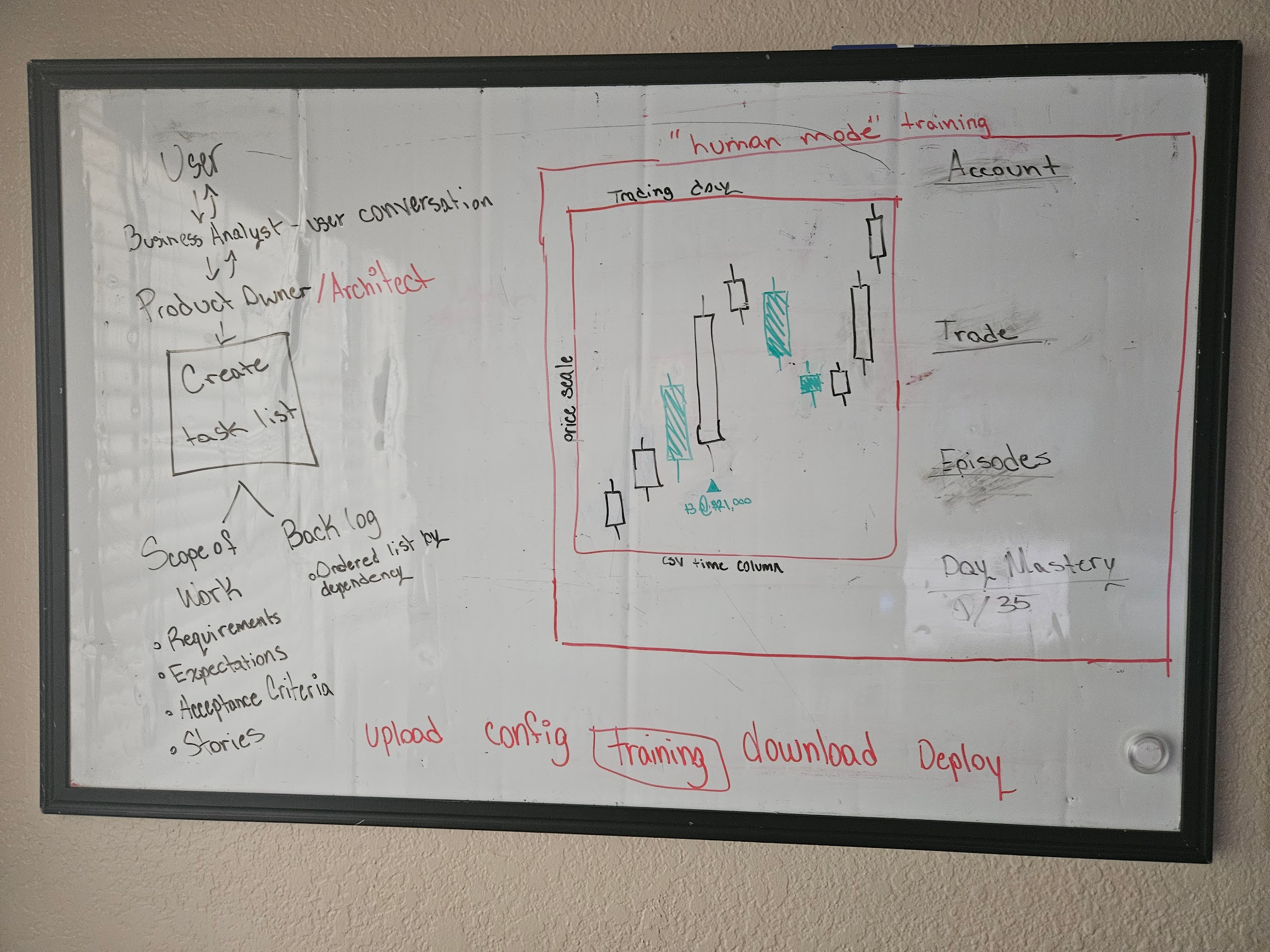
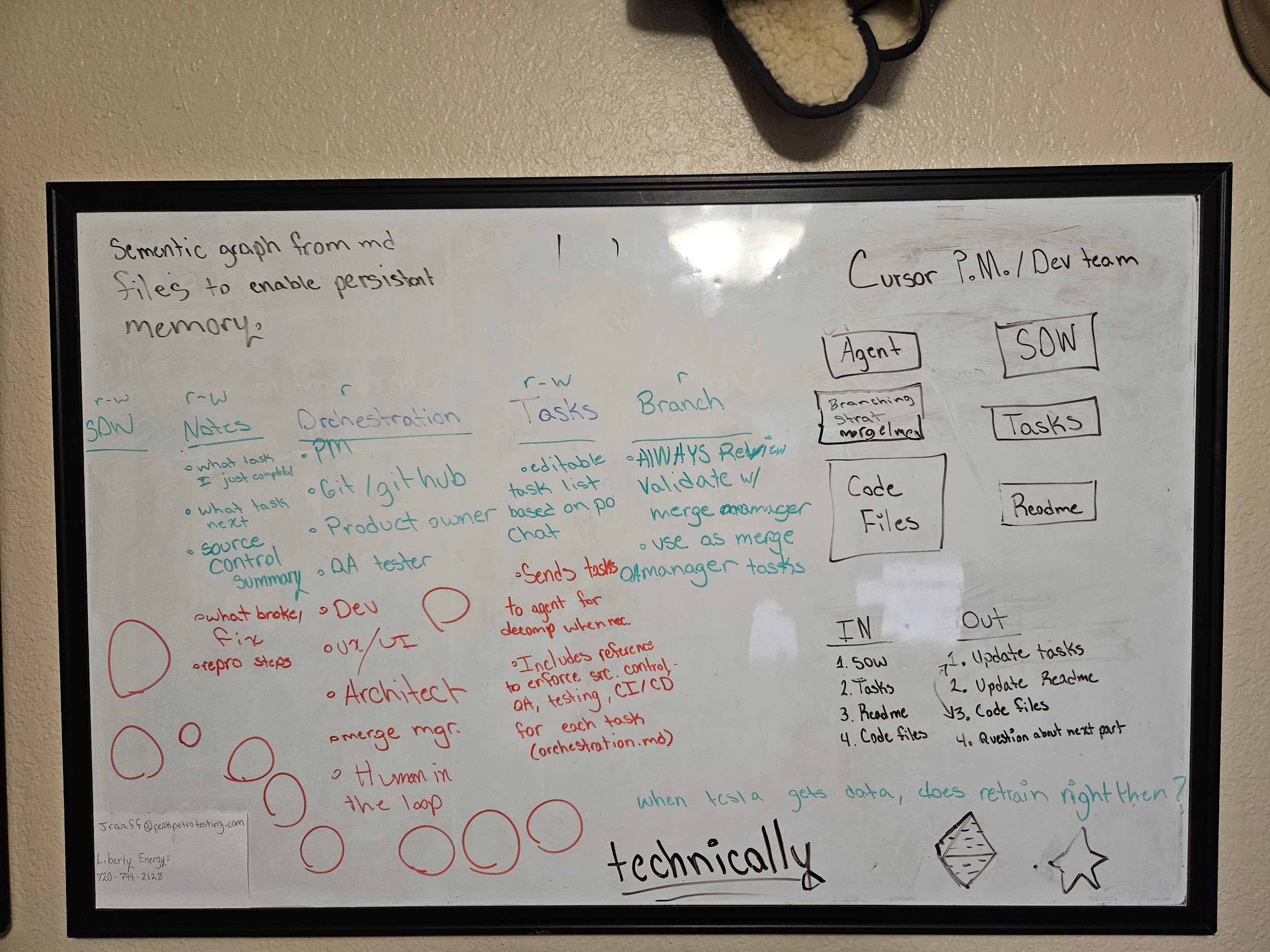
Focused Risk Management The live trading module must include essential risk controls:

* Daily loss limits
* equity based position sizing limits
* enhanced kill-switch functionality
* Core compliance logging
* Docker Containerization
* Dockerfile for simulation/training
* Docker Compose for local development
* Enhanced Visualization
* web dashboard for monitoring training
* visualization of trading performance

“Human” Rendering training mode:  
Displays:  
1. Account Metrics

2. Trade Metrics Per Episode

* Total Trades



Statement of work