Tab 1

**Statement of Work**

**Project Title:** Reinforcement Learning Futures Trading System

**Objective:** The objective of this system is to train an agent (P.P.O.) to consistently achieve a configurable daily profit target by using historical futures price data to create a high fidelity simulator and master individual trading days through a meta day mastery mechanism and transfer learning, and finally deploying for live trading through the Tradovate API with robust risk controls after training is complete. The solution will deliver a full stack desktop application with an advanced UI/UX, a production-grade system that includes an high fidelity simulator environment with proper design, reward mechanisms, day mastery protocols, and a PPO agent that has a seamless transition from simulation to live trading.

**Scope of Work:**

**Phase 1: Core Environment & Data Ingestion**

This phase focuses on building the foundational components of the system: the data pipeline and the high-fidelity simulation environment. This is the most critical phase, as the quality of the training data and the realism of the simulator directly impact the agent's performance.

**Deliverables:**

1. **Data Ingestion Module:**
   1. A robust module capable of reading historical futures data from a user-provided CSV or Excel file.
   2. The module cleans the data by rejecting days with missing values, and validates the number of days in the data set
   3. The module will parse the data, splitting it into individual trading days based on configurable start and end times (e.g., 1-minute candles from 9:00 AM to 4:00 PM CST).
   4. This module will act as a streaming simulator, feeding one day's worth of data at a time to the training environment.
2. **Gymnasium RL Environment (TradingDataEnv):**
   1. A custom-built gymnasium-compatible environment that simulates futures trading with high fidelity.
   2. **State Space:** The observation space will include stacked history of OHLC data, essential technical indicators (e.g., EMA, SMA, Bollinger Bands, ATR, MACD), and real-time account metrics (equity, P/L, position information).
   3. **Action Space:** The agent will have four discrete actions: "Buy Max", "Sell Max", "Hold", and "Close All".
   4. **Realistic Market Mechanics:** The environment will accurately model:
      1. **OHLC/High-Low P/L Calculation:** Realistic profit and loss calculations based on the intraday high/low prices, not just the close.
      2. **Commission & Slippage:** Configurable fees and slippage to accurately reflect real-world trading costs.
      3. **Margin Enforcement:** Strict enforcement of margin requirements, with a configurable equity reserve threshold to prevent bankruptcy.
   5. **Reward Function:** The primary reward will be based on the change in net equity, with a significant bonus for hitting the configurable daily profit target. Configurable penalties will be applied for commissions, high trade count, and excessive drawdown.
   6. **Termination Conditions:** An episode will terminate upon hitting the daily profit target, reaching a configurable daily loss limit, or the end of the trading day.

**Phase 2: Training Pipeline & Day Mastery**

This phase integrates the environment with a robust training pipeline, implementing the core machine learning logic and the unique day mastery mechanism.

**Deliverables:**

1. PPO Training Pipeline:
   * The PPO agent will be implemented using the Stable-Baselines3 library.
   * Hyperparameter Tuning: A configuration file will allow for easy adjustment of hyperparameters.
   * Checkpointing: The model's weights and training state will be saved periodically, allowing for training to be resumed.
   * Structured Logging: A comprehensive logging system will track all relevant training metrics (profit target hit rate, win rate, max drawdown, etc.).
2. Day Mastery & Transfer Learning Mechanism:
   * The system will track per-day success metrics.
   * A day will be considered "mastered" when the agent's performance meets the configurable criteria:
     + A minimum of X episodes have been run (defaulting to 1000).
     + The profit target has been hit in at least 95% of those episodes.
     + Performance has plateaued with no significant improvement over Y episodes (defaulting to 100).
   * Upon mastering a day, the system will save the model's weights and begin training on a new, unmastered day. This transfer learning approach allows the agent to build knowledge from a diverse set of market conditions.

#### **Phase 3: Desktop Application & Visualization**

This phase is dedicated to building the user interface, providing a complete desktop application for managing the entire process.

**Deliverables:**

1. **Full-Stack Desktop Application (UI/UX):**
   * A unified application built with a framework, providing a cohesive user experience.
   * **Upload/Settings Page:** A clean interface for users to upload their historical data file and configure all system parameters (profit targets, risk limits, day mastery criteria, etc.).
   * **Training Page:** A page with two rendering modes:
     + **"Human" Rendering Mode:** Displays a live, animated candlestick chart of the simulated data stream. The agent's actions (buy/sell/close) will be plotted directly on the chart. A real-time dashboard will show critical metrics (account balance, P/L, trade count, day mastery status). This mode is for visual inspection and debugging.
     + **"Fast" Rendering Mode:** An ultra-fast training mode that disables all graphical rendering and logging to the screen, showing only a simple progress bar. All data will be saved to logs for post-training analysis.
2. **Post-Training Analysis & Model Management:**
   * Once training is complete, the application will allow the user to download the final trained model files.
   * Users will also have the ability to download a partially trained model if training is stopped prematurely.

#### **Phase 4: Live Trading Deployment & Containerization**

This final phase focuses on the transition from simulation to live trading, ensuring the system is production-grade, secure, and reliable.

**Deliverables:**

1. **Tradovate API Connector:**
   * A dedicated module to connect to the Tradovate API for live market data streaming and order management.
   * The connector will preprocess real-time OHLC data from the API to match the format used during training, ensuring a seamless transition for the agent.
   * The agent's actions (Buy Max, Sell Max, etc.) will be correctly translated and executed as API commands.
2. **Live Trading Dashboard:**
   * A dedicated page within the desktop application for monitoring live trading.
   * Displays real-time account metrics (current balance, daily P/L, margin usage), a history of executed trades, and a live log of all API communication.
   * **Risk Controls:** The dashboard will prominently feature and enforce the following:
     + **Configurable Daily Maximum Loss Limit:** An automated kill-switch that will immediately close all positions and cease trading for the day if the loss limit is reached.
     + **Position Sizing:** The system will dynamically calculate and enforce position sizing based on the user-configured contract value and account balance.
   * **Emergency Kill-Switch:** A prominent, one-click "kill-switch" button to manually close all open positions and halt all trading activity.
3. **Containerization (Docker):**
   * A Dockerfile will be created to containerize the entire application.
   * Docker will be used to ensure a consistent, reproducible environment for both development and local deployment on Windows, eliminating potential dependency conflicts.
4. **Documentation & Testing:**
   * **Technical Documentation:** Comprehensive documentation for all components, including the environment, training pipeline, and Tradovate API connector.
   * **Configuration Guide:** A guide explaining all configurable parameters.
   * **Comprehensive Testing:**
     + **Unit Tests:** For core functions like P/L calculation, margin enforcement, and indicator logic.
     + **Integration Tests:** To ensure the training pipeline and API connector work together seamlessly.
     + **Stress Testing:** To validate the system's resilience under heavy data loads and edge-case scenarios.

**Milestone Demonstrations:**   
Phase 1:

* Demonstrate the data ingestion module's ability to successfully parse a user-provided CSV/Excel file, split it into trading days, and stream it to the environment.
* Run the TradingDataEnv in a standalone simulation mode, proving that it accurately calculates P/L, enforces margin, and handles configurable commissions and slippage. Show that the state space is correctly populated with market data, indicators, and account metrics.

Phase 2:

* Show a complete training session where the agent trains on one day, successfully "masters" it according to the configurable criteria, and then seamlessly transitions to training on a second day, demonstrating the transfer learning and day mastery mechanisms.
* Present the training logs and a simple console dashboard showing real-time metrics, proving that the system is tracking performance correctly.

Phase 3:

* Run the desktop application and demonstrate a full training session using both "Human" and "Fast" rendering modes.
* Show how to upload a data file, configure all parameters, start training, and then download the final trained model files.

Phase 4:

* Connect the system to the Tradovate API in a paper-trading environment.
* Show real-time data streaming and how the agent's actions are correctly executed.
* Demonstrate the risk controls in action, specifically the automatic daily loss limit and the manual kill-switch.
* Show the live trading dashboard, its logging capabilities, and the seamless transition from a local model to a live trading agent.
* Provide the Docker container, proving a reproducible build of the entire application.

OLD VERSION::  
**Scope of Work:**

**Phase 1:**

1. Collect CSV or Excel file from the user (*Only one file is allowed*) that contains historical price data
2. File data is read chunk by chunk, Parsed into individual trading days for use in the environment to mimic a live API feed [1]
3. Load days into a flexible CSV to stream and begin streaming to the preprocessing mechanism in the environment [2]
4. The environment preprocesses the incoming streamed price data, calculating and indicators and all time-frames to be used as observations in the state space and rendered if human mode is enabled [3]

**Phase 2:**

1. The environment is custom built using gymnasium to step through the data one row at a time, do any necessary calculations and then use as observations for the agent
2. The agent outputs one of four (4) possible actions: Buy Max, Sell Max, No operation (Hold) and Close All
3. The agent trains in a robust pipeline that is controlled by the day switching mechanism and features proper episode termination conditions, day mastery and transfer learning
4. Wa
5. The
6. Place holder

**Phase 3:**

1. The app Supports “Human” and “fast” rendering modes for training where:
   1. Human rendering mode displays or plots the observation space in the application user interface and must be complete with:
      1. Current Episode Account Metrics
      2. Current Trade metrics (Expected in the observation space)
      3. Most recent trade metrics
      4. Day mastery metrics
      5. Current Day Episode Metrics
      6. Cumulative Episode metrics
      7. Plots the incoming price data stream at a configurable rate (e.g., X candles sent per minute)
      8. Switchable timeframe view
   2. Fast rendering that allows for an ultra fast training mode by:
      1. Showing ONLY a Progress bar
      2. Keeping comprehensive logs saved for review after training
2. g
3. Place holder
4. Place holder

**Timeline:**

**Milestone Demonstrations:**   
Phase 1:

* Demonstrate the data ingestion module's ability to successfully parse a user-provided CSV/Excel file, split it into trading days, and stream it to the environment.
* Run the TradingDataEnv in a standalone simulation mode, proving that it accurately calculates P/L, enforces margin, and handles configurable commissions and slippage. Show that the state space is correctly populated with market data, indicators, and account metrics.

Phase 2:

Phase 3:



**User Benefits:**

* F

[1] Parse historical price data into individual trading days for modeling the problem as the mastery of reaching the profit target during *individual* trading days and transfer learning (*start and end times configured in a settings page*)

[2] Load random trading days (*Based on Day mastery conditions/ controlled by day mastery mechanism*) into a flexible CSV to stream and begin streaming to the preprocessing mechanism in the environment (*there will be a preprocessing mechanism to facilitate live trading*)

[3] The environment preprocesses the incoming streamed price data (*time, Open, High, Low, and Close, using individual rows from each day in the file*) calculating and normalizing/scaling indicators and all time-frames (*higher than the timeframe of the price data uploaded by the user*) to be used as observations in the state space (*all* *calculated time frames and indicator values for all time frames available*)