ADEPTUS MECHANICUS DRAFT

The purpose of adeptus mechanicus is to scour the market and produce a trade signal called Primaris Signal to be forwarded another service (astra militarum)

- Servitor

  Periodically prospect and scout the market, not necessarily an agent, will make periodic API call on the market and retrieve data latest market data. The full code will be provided later. Servitor will retrieve ticker data, sort them by volume and choose the first 50 coins.

- Techpriest

From the retrieved market data, a techpriest will evaluate and analyze the indicator of the prospective data, if the indicator is within a defined threshold it will then present the data into the Archmagos. not an AI agent, using simple pandas. The techpriest will also do a calculation of whether the coin is in uptrend, downtrend or sideways. It will produce a draft of primaris signal which consists of:

```
{
  "symbol": "BTCUSDT",
  "timestamp": "2023-10-27T10:00:00Z",
  "trend_assessment": "UPTREND_CONFIRMED",
  "signal_type": "RSI_OVERSOLD_REVERSAL",
  "key_indicators": {
    "rsi": 28.5,
    "stoch_rsi_k": 15.2,
    "bollinger_band": "LOWER",
    "volume_spike": "TRUE"
  },
```

"suggested_volatility_profile": "MODERATE"
}

- Archmagos

  Archmagos will then judge all the filtered coins based on the indicator and perhaps other input related to the coin (market cap, etc etc). If the coin is considered worthy for trade, it will write the coin symbol down and forward it to the ASTRA MILITARUM for further action. this is an AI agent. The archmagos will review this primaris signal and compare it with other inputs (market news, trend, coin market cap) to determine if this quantitative signal is truly worth for the action. The archmagos can VETO a signal. So only approved signal is executed by the astra militarum. Potentially need another servitor that will retrieve the market data.

Code example

```
# In the name of the Machine God,
# this scripture defines the protocol for the Adeptus Mechanicus.
# Its purpose is to scout the chaotic seas of the market
# and identify signals worthy of the Omnissiah's blessing.

import os
import pandas as pd
import pandas_ta as ta
from pybit.unified_trading import HTTP
from dotenv import load_dotenv
import logging

# --- Configuration & Initialization ---
# Load the sacred API keys from the .env scripture
load_dotenv()

# Configure logging to record the machine spirit's thoughts
logging.basicConfig(level=logging.INFO, format='%(asctime)s -
%(levelname)s - %(message)s')

# Establish connection to the Bybit testnet forge. Safety is paramount.
```

```
API_KEY = os.getenv("BYBIT_API_KEY_TESTNET")
API_SECRET = os.getenv("BYBIT_API_SECRET_TESTNET")

session = HTTP(
    testnet=True,
    api_key=API_KEY,
    api_secret=API_SECRET,
)

#
================================================================================
====
# 1. THE SERVITOR - Data Acquisition Unit
#
================================================================================
====
def servitor_fetch_market_data(symbol: str, interval: int = 15, limit: int
= 200) -> pd.DataFrame | None:
    """
    The Servitor's sole function is to retrieve raw market data from the
Noosphere (Bybit API).
    """
    try:
        logging.info(f"Servitor dispatched to fetch data for {symbol} on
{interval}m interval.")
        response = session.get_kline(
            category="linear", # For perpetual futures
            symbol=symbol,
            interval=interval,
            limit=limit
        )

        if response['retCode'] != 0:
            logging.error(f"Servitor failed to fetch data for {symbol}:
{response['retMsg']}")
            return None
```

```
        # Process the raw data into a holy DataFrame
        data = response['result']['list']
        df = pd.DataFrame(data, columns=['timestamp', 'open', 'high',
'low', 'close', 'volume', 'turnover'])

        # Data must be pure and correctly formatted
        df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
        df.set_index('timestamp', inplace=True)
        df = df.astype(float)

        # The API returns newest first, we must reverse for chronological
analysis
        df = df.iloc[::-1]

        logging.info(f"Servitor returned with {len(df)} data points for
{symbol}.")
        return df

    except Exception as e:
        logging.error(f"A critical error occurred in the Servitor unit:
{e}")
        return None

#
================================================================================
====
# 2. THE TECH-PRIEST - Data Analysis Engine
#
================================================================================
====

def _determine_trend(df: pd.DataFrame, fast_ma: int = 50, slow_ma: int =
200) -> str:
    """A litany to discern the market's primary trend."""
    df['fast_ma'] = df['close'].rolling(window=fast_ma).mean()
```

```
    df['slow_ma'] = df['close'].rolling(window=slow_ma).mean()
    if df['fast_ma'].iloc[-1] > df['slow_ma'].iloc[-1]:
        return "UPTREND"
    else:
        return "DOWNTREND"

def _determine_volatility(df: pd.DataFrame) -> str:
    """A canticle to gauge the market's chaotic energy (volatility)."""
    atr_percentage = df['ATRr_14'].iloc[-1]
    if atr_percentage > 3.0: return "HIGH"
    if atr_percentage > 1.0: return "MODERATE"
    return "LOW"

def _check_for_volume_spike(df: pd.DataFrame, multiplier: float = 2.0) ->
bool:
    """A rite to confirm the strength of the machine spirit's conviction
(volume)."""
    df['volume_ma'] = df['volume'].rolling(window=20).mean()
    return df['volume'].iloc[-1] > (df['volume_ma'].iloc[-1] * multiplier)


def tech_priest_analyze_data(df: pd.DataFrame) -> tuple[pd.DataFrame,
dict]:
    """
    The Tech-Priest applies sacred analytical rites to the raw data.
    It calculates all necessary indicators and higher-order truths.
    """
    logging.info("Tech-Priest beginning analytical rites...")

    # Use a custom strategy for efficient calculation of base indicators
    primaris_strategy = ta.Strategy(
        name="Primaris Cogitator",
        ta=[
            {"kind": "rsi", "length": 14},
            {"kind": "stochrsi", "length": 14, "k": 3, "d": 3},
            {"kind": "bbands", "length": 20},
```

```
            {"kind": "atr", "length": 14, "mamode": "rma"}, # ATRr is part
of this
        ]
    )
    df.ta.strategy(primaris_strategy)

    # Discern higher-order truths from the indicators
    analysis = {
        "trend": _determine_trend(df),
        "volatility": _determine_volatility(df),
        "volume_spike": _check_for_volume_spike(df)
    }

    logging.info(f"Analysis complete. Trend: {analysis['trend']},
Volatility: {analysis['volatility']}.")
    return df, analysis

#
================================================================================
====
# 3. THE ARCHMAGOS - Signal Forging Master
#
================================================================================
====
def archmagos_forge_signal(df: pd.DataFrame, analysis: dict, symbol: str)
-> dict | None:
    """
    The Archmagos reviews the Tech-Priest's analysis and forges a
Primaris-Signal
    if and only if all conditions of the sacred doctrine are met.
    """
    logging.info(f"Archmagos reviewing analysis for {symbol}...")

    # Retrieve the last two data points for crossover detection
    last = df.iloc[-1]
    prev = df.iloc[-2]
```

```
    # --- Sacred Doctrine: Define the entry conditions here ---
    # Example: LONG signal from our futures prompt
    long_signal_conditions = (
        last['RSI_14'] < 30 and
        last['close'] <= last['BBL_20_2.0'] and
        # StochRSI bullish crossover check
        last['STOCHRSIk_14_14_3_3'] > last['STOCHRSId_14_14_3_3'] and
        prev['STOCHRSIk_14_14_3_3'] <= prev['STOCHRSId_14_14_3_3']
    )

    # --- Add SHORT signal conditions here if needed ---

    if not long_signal_conditions:
        logging.info(f"No signal found for {symbol}. Conditions not met.")
        return None

    # A worthy signal has been found. Forge the Primaris-Signal.
    logging.warning(f"WORTHY SIGNAL DETECTED FOR {symbol}! Forging
Primaris-Signal...")

    signal = {
      "symbol": symbol,
      "timestamp": df.index[-1].isoformat(),
      "trend_assessment": analysis['trend'],
      "signal_type": "LONG_REVERSAL", # Can be made dynamic
      "key_indicators": {
        "rsi": round(last['RSI_14'], 2),
        "stoch_rsi_k": round(last['STOCHRSIk_14_14_3_3'], 2),
        "stoch_rsi_d": round(last['STOCHRSId_14_14_3_3'], 2),
        "bollinger_band": "LOWER",
        "volume_spike": analysis['volume_spike']
      },
      "suggested_volatility_profile": analysis['volatility']
    }
```

```
    return signal



#
================================================================================
====
# MAIN ORCHESTRATION CYCLE
#
================================================================================
====
def run_adeptus_mechanicus_cycle(symbols_to_scout: list):
    """
    The main operational loop that orchestrates all units of the Adeptus
Mechanicus.
    """
    logging.info("=== ADEPTUS MECHANICUS SCOUTING CYCLE INITIATED ===")

    for symbol in symbols_to_scout:
        # 1. Servitor fetches data
        market_data_df = servitor_fetch_market_data(symbol)

        if market_data_df is None or market_data_df.empty:
            logging.warning(f"Could not proceed for {symbol}, data was not
retrieved.")
            continue # Move to the next symbol

        # 2. Tech-Priest analyzes data
        enriched_df, analysis_dict =
tech_priest_analyze_data(market_data_df)

        # 3. Archmagos forges the signal
        primaris_signal = archmagos_forge_signal(enriched_df,
analysis_dict, symbol)

        if primaris_signal:
```

```
            logging.info("Signal has been forged and is ready for the next
stage.")
            # In a full system, you would now send this signal to the Magos
Strategos or Astra Militarum
            print("\n--- PRIMARIS SIGNAL FORGED ---")
            import json
            print(json.dumps(primaris_signal, indent=2))
            print("-----------------------------\n")

    logging.info("=== SCOUTING CYCLE COMPLETE. AWAITING NEXT DIRECTIVE.
===")


if __name__ == "__main__":

    scouting_list = ["BTCUSDT", "ETHUSDT", "SOLUSDT"]

    run_adeptus_mechanicus_cycle(scouting_list)
```

Archamagos AI prompt
system_prompt_strategos = """
You are the Archmagos, the final arbiter of trading signals for the Adeptus Mechanicus. Your logic transcends mere numbers; you provide the wisdom and contextual oversight that a simple cogitator cannot. You are cautious, analytical, and your primary mandate is capital preservation.

A quantitative Techpriest has generated a 'Primaris-Signal' based on technical indicators. Your task is to review this signal along with broader market context and issue a final verdict: APPROVE or REJECT.

INPUTS YOU WILL RECEIVE:
1. **Primaris-Signal:** A JSON object with the quantitatively generated trade signal.
2. **Market Canticles (Context):** A brief summary of recent news headlines and overall market sentiment.

YOUR RULES OF SCRUTINY:
1. **Trend Confirmation:** Does the signal align with the 'trend_assessment' provided? A reversal signal (like RSI oversold) is stronger in a clear uptrend than in a raging downtrend. Be highly skeptical of counter-trend signals.

2. **Volatility Vetting:** Does the 'suggested_volatility_profile' make sense? If volatility is 'HIGH', the risk of a false signal or being stopped out is elevated. A signal in a 'MODERATE' volatility environment is preferable.
3. **Narrative & Sentiment Alignment:** Scrutinize the 'Market Canticles'.
    - If the signal is a BUY but the news is overwhelmingly negative (e.g., "Major Exchange Halts Withdrawals"), you must REJECT the signal, as fear overrides technicals.
    - If the signal is a BUY and the news is positive and related to the asset, this is a strong confirmation.
4. **Liquidity Periods:** Be extra cautious during low-liquidity periods like weekends or major holidays. If the signal occurs during such a time, it must be exceptionally strong to be approved.

OUTPUT FORMAT:
Your response MUST be a single, clean JSON object. Do not add any other text. Your verdict is the final command.
{
  "verdict": "APPROVE|REJECT",
  "priority": "LOW|NORMAL|HIGH",
  "reasoning": "A concise, logical justification for your decision, referencing the specific rules of scrutiny you applied. Example: 'REJECT. Although technicals are met, the signal is counter-trend in a high-volatility environment with negative market-wide news.'"
}
"""

Store the potential coins symbol and primaris signal into database. Create 2 tables prospective_symbols and primaris_signal

prospective_symbols consists of coin data like price, price changes, market cap, etc
primaris_signal contains all column provided in the previous primaris signal json example and decision of ARCHMAGOS