Governance Proposal Trading Framework

1. Technical Infrastructure Setup

1.1 Forum Ingestion & Monitoring

1. **Data Source Identification**
    - Create a **comprehensive registry** of the following: (1) crypto governance; (2) Twitter handles; (3) Twitter handles of the founder(s); and (4) the blog page for assets listed on the relevant exchanges where the trading firm is active. A package such as CCXT can quickly provide a list of assets. A researcher on the team will need to compile this registry manually but will be assisted by services such as Blockworks' GovHub and Messari in this compilation process. In addition, for each relevant protocol, the analyst should add a link to the relevant website, which links the proposals once they have entered the on-chain voting stage and tracks the voting results (this is often hosted on a proprietary website or a website such as Tally.
    - The analyst and PM can (before adding the additional detail) also remove any assets deemed non-tradable for whatever reason
    - This registry should include direct URLs, forum types (Discourse, Reddit, custom websites), and whether an API or RSS feed is available.
    - This registry will then be converted to a JSON file, which can continuously be updated over time through a mix of periodically running a script (once a day) to check new assets have been listed and then the analyst added the relevant detail
2. **Scraping & Polling Mechanisms**
    - **Automated Bots**: Write dedicated scrapers or use existing tools (e.g., BeautifulSoup, Selenium) to parse each forum. For platforms with RSS or API feeds, ingest the feed every few seconds to one minute. Since Discourse runs most crypto governance platforms, the API documentation will likely be helpful here. In addition, the Mintscan API will be valid for Cosmos SDK-based governance (once they have been submitted on-chain)
    - **API Integration**: For forums offering API endpoints or RSS feeds, query them every 30 seconds for new discussion threads or proposals[1].
    - **Scalability**: Set up multiple scrapers running in parallel microservices to handle spikes in posts and ensure minimal delay in detection.

[1] Only check for new threads or proposals and not new comments or replies in existing threads

3. **Timestamp & Queuing**
    - Once a new thread is detected, **record the exact timestamp** of discovery.
    - Place the post data (title, author, timestamp, forum link) into a “New Posts” message queue (e.g., RabbitMQ, Kafka) for immediate downstream processing.
4. **Failover & Redundancy**
    - Implement a **redundant queue** or fallback ingestion process. For instance, if the primary job fails, a secondary job checks the forum slightly less frequently. This prevents critical posts from being missed due to a single point of failure.

1.2 Database & Storage

1. **Structured Repository**
    - Store core metadata (post ID, timestamp, title, link, forum name, author) in a relational database for **fast queries**.
    - Maintain indexing on crucial fields (asset name, timestamp) to speed up retrieval.
2. **Raw Text Storage**
    - If advanced text querying or full-text search is needed, save the full text and any attachments or links from each post in a document store (e.g., Elasticsearch or MongoDB).
3. **Data Consistency & Archiving**
    - If the forum software allows edits, keep a **versioned history** of post updates. This ensures you can track changes in governance proposals over time.

1.3 Notification & Alerting System

1. **Event Trigger**
    - When a new post enters the queue, send a **real-time event** to the pipeline orchestrator (e.g., Celery, AWS Lambda).
    - Tag the post as “unprocessed” until an analysis is complete.
2. **Push Alerts**
    - Use a **WebSocket** or server-sent event approach to push notifications to a custom Telegram Group chat (or front-end dashboard if the engineering resources are available) so the researcher sees new unprocessed posts in near real-time.
    - The relevant asset ticker, the title of the post, the author, and a link to the full post will be initially provided to the analyst. Further summarisation will be provided in the next step (with the required delay for both NLP & LLM analysis to occur)
    - If a critical keyword is detected using a simple regex query (e.g., “hack,” “security exploit,” “emergency governance,” “vote passed”), optionally trigger a **high-priority** push alert (e.g., SMS, Slack, or instant messaging).

- In addition, if there are any keywords in the body of the post, this will trigger a high-priority push alert (e.g. 'buyback', 'burn', 'mint', 'supply', 'inflation', 'deflation', 'revenue', 'allocation', 'dividend', 'yield')
- We carry out NLP & LLM analysis at the next step to minimise the time from which the analyst first receives an alert

---

2. NLP & LLM Analysis

2.1 Initial Classification & Filtering

1. **Keyword/Pattern Matching**
   - Scan text for key terms like “proposal,” “vote,” “emergency,” “tokenomics,” “fee change,” or references to actual parameter adjustments (e.g., interest rates, reward distribution). In addition to the keywords mentioned above ('buyback', 'burn', 'mint', 'supply', 'inflation', 'deflation', 'revenue', 'allocation', 'dividend', 'yield').
   - If these are found, immediately label the post as *potentially necessary*, even before deeper analysis.
2. **Machine Learning Classifier**
   - Use a **lightweight classification model** (e.g., logistic regression or a fine-tuned Transformer) to assign a numeric importance score.
   - Factors: the presence of keywords, engagement metrics (like the number of comments/views within minutes), and historical context (e.g., how often a similar post led to market movement).
3. **Priority Queue Update**
   - Posts with high importance scores move to the *priority queue* for immediate deep analysis, while others go to a standard queue.
   - If the text length is minimal (like a short question or trivial note), it may be deprioritised automatically.

2.2 Deep Analysis with LLMs

1. **Summarisation Pipeline**
   - Send the post’s text to an **LLM-based summariser** (the GPT-4.5 model would be appropriate here, given its speed and strong linguist ability compared to the reasoning models).
   - We would use the following system prompt and feed the model the post title and body alongside relevant metadata (datetime posted, author name, etc.).
   - The system prompt would be as follows:

> **System/Role Instructions**
> You are a financial analyst specialising in cryptocurrency price movements. You will receive text from a crypto governance forum discussing a proposed change or update to a specific token or protocol. Your primary objective is to analyse the discussion and produce a concise summary—suitable to be read in under one minute—highlighting how the proposal may affect the token’s price via supply or demand dynamics.

User Prompt

1. **Context**: I will provide the text from a crypto governance forum below.
2. **Task**:
    - Identify any proposed mechanisms that could increase or decrease the token's supply (e.g., token minting, burn, inflation changes, lockups).
    - Identify any proposed mechanisms that could raise or lower demand (e.g., new incentives for buying, new utilities, dividend/buyback, staking rewards, or changes in adoption).
    - Note any "mechanical" changes to buying or selling behaviour that might shift flow (e.g., forcing certain entities to buy or sell, introducing automatic buybacks, or enabling large token unlocks).
    - Briefly assess whether and how these changes might move the price up or down.
    - Keep the summary as short as possible without omitting critical facts—ideally, in a single paragraph or a few concise bullet points.
3. **Output Style**:
    - Begin with a **one-sentence** summary of the overall market impact (positive/negative/uncertain).
    - Use **clear, direct language**.
    - Summarise any significant supply/demand changes in **bullet points** or a **short paragraph** that fits within a 60-second read.
    - Do **not** include lengthy explanations, disclaimers, or secondary details.
    - Do **not** repeat or quote large text sections; **only** provide essential insights.
4. **Forum Text**:
    - Place the exact text from the governance forum here.
5. **Sample Output Format (Hypothetical Example)**:
    - *"Overall Impact: Slightly bullish.*
      • The proposal reduces circulating supply through a moderate token burn schedule.
      • New staking rewards might encourage short-term buying and holding.
      • Likely short-term price boost from reduced sell pressure, though the long-term impact is uncertain."

Ensure your final response is **no longer than 100–150 words** or **3–5 bullet points**. Then, conclude the summary.

2. **Sentiment & Impact Analysis**
    - Pass the text to a **separate LLM** or an internal sentiment model trained on governance-related data. Ask it to label the post as Bullish, Bearish, or Neutral in the context of the associated asset.
    - Optionally, add a confidence score (0-1) indicating how confident the model is of its sentiment label.
3. **Confidence Threshold & Escalation**
    - If the model output is uncertain (low confidence) or if the language is ambiguous, automatically flag it for human re-check or second-model pass.
    - Elevate the alert priority further for extremely critical or unusual proposals (e.g., significant changes in protocol fees).

2.3 Output Formatting

1. **JSON Structuring**
    - Combine all the results (importance score, LLM summary, sentiment label, references) into a single JSON object.
    - Example:

```
JSON
CopyEdit
{
  "post_id": "abc123",
  "asset": "UNI",
  "importance": "High",
  "summary": "This proposal suggests activating the fee switch on Uniswap v3
pools...",
  "sentiment": "Bullish",
  "confidence": 0.88,
  "timestamp": "2025-03-24T10:15:00Z"
}
```

2. **Database Insert**
    - Insert this structured output back into the database so it's available for the front-end dashboard or will be pushed to the Telegram Group Chat.
    - Update the post's status from "unprocessed" to "processed."

---

3. Researcher Dashboard & Interface

3.1 Real-Time Feed

1. **Live Updating Dashboard**
    - Display a chronological list of processed posts, with each item showing:
        - **Asset** (logo + name)
        - **Proposal Title** or short description
        - **Importance** (High/Medium/Low)
        - **Sentiment** (Bullish/Bearish/Neutral)
        - **Timestamp** (time since the post was made)
    - Implement auto-refresh or WebSocket push so new items appear instantly without manual page refresh.
    - This data will also be pushed at the same time to the Telegram Group Chat with push notifications for high-importance posts
2. **Filtering & Searching**
    - Provide a top bar with filters:
        - Importance level (check boxes for High, Medium, Low)
        - Sentiment (Bullish/Bearish/Neutral)
    - The search bar allows the researcher to type keywords to quickly find historical posts (e.g., "fee switch").

3.2 Detailed View

1. **Side Panel/Modal**
    - When the researcher clicks an alert, it opens a **detailed view** showing:
        - **LLM Summary** in bullet points
        - **Sentiment & Confidence**
        - **Author & Forum Link** (with a button to open the original post)
        - **Relevant Historical Proposals** or reference links
2. **Market Data Snapshot**
    - (Optional but recommended) Display a **mini chart** of the asset's real-time price over the last hour or day.
    - Show key trading metrics: current price, 24-hour volume, percentage change, and order book depth if integrated.
    - It can simply be an embedded Trading View-esque chart. See Velo's example.
    - This would help the researcher understand to what extent the market has already ingested the proposal.

3. **Notes & Annotations**
    - Allow the researcher to add a quick text note (e.g., "Suspect immediate price impact, watch volume.")
    - Store these notes for future reference or compliance/audit.

---

4. Human Researcher Decision-Making

4.1 Alert Arrival & Triage

1. **Immediate Action on High-Priority**
    - If an alert is flagged as High importance (e.g., critical security fix, major tokenomics shift), the researcher checks it first.
    - The sentiment label (Bullish/Bearish) provides a **directional hint** for potential trade and can either reply in Group Chat or on the Dashboard to convey investment ideas as a result
    - Researcher can also then tag the given forum thread to receive notifications (and similar NLP and LLM analysis notifications for future replies on the thread)
2. **Multi-Alert Collision Handling**
    - If multiple high alerts arrive simultaneously (e.g., from different protocols), the researcher quickly scans which one is **most time-sensitive** (security exploit vs. governance vote that ends in a week).

4.2 Deeper Investigation

1. **Original Post & Community Feedback**
    - If details are unclear or the post has multiple layers, the researcher may open the original forum link to skim the actual discussion or see community replies (if available).
    - They can also check other sources, such as Twitter or other venues they know, to review proposals and their current state if they wish to gauge broader sentiment or need to clarify specific details

4.3 Trade Execution

1. **Decision Criteria**
    - The researcher applies their discretionary strategy to evaluate whether the news is market-moving or incremental.
    - The researcher aims to give snapshot Long, Short, and No-trade suggestions along with Conviction (Low, Mid, High), which will correspond to a trade size at the PM's discretion
    - Ultimately, trade decision is made by the PM based on investment analyst snapshot judgement, or the PM can decide independently based on the summary
2. **Documentation & Logging**
    - In the case that a trade is made, the trade idea, the corresponding alert that triggers it, then the outcome of the trade is logged
    - This log is stored for compliance and performance analysis.

---

5. Monitoring & Continuous Improvement

5.1 Performance Tracking

1. **Latency Metrics**
    - Track **time stamps** from forum post creation to detection, from detection to classification, and from classification to researcher notification.
    - Analyse average and 95th percentile latencies monthly. Aim to keep total pipeline time under 1-2 minutes.
2. **Classification Accuracy**
    - Maintain a record of how often "High importance" posts correlate with actual market movement or a trading decision.
    - Refine LLM prompts if too many false positives or false negatives occur.
3. **Sentiment Accuracy**
    - Compare sentiment labels with actual short-term price movements or researcher feedback. If the sentiment label is consistently off, retrain the NLP model.

5.2 Researcher Feedback Loop

1. **Qualitative Annotations**
    - The researcher can mark alerts as “Irrelevant,” “Useful but no trade,” or “Trade executed.” This helps refine the model.
    - For example, if an alert was labelled Bearish but the researcher concluded it was Bullish, that data can improve the classifier.
2. **Model Retraining**
    - Periodically retrain your classification and sentiment models using the newest forum data plus user feedback.
    - If using an LLM API, update prompts or fine-tune the model with new examples of correct governance post interpretation.

5.3 Trading Outcome Attribution

1. **PnL Attribution**
    - Tag trades in the firm’s booking system with a “governance-based alert” label.
    - Aggregate monthly: “Governance-based trades netted $X, with Y% success rate.”
2. **Refine Strategy**
    - If specific governance proposals consistently yield profitable trades (e.g., “fee changes in stablecoin protocols”), the researcher might want to scale up the strategy in terms of sizing
3. **Comparative Benchmarks**
    - Compare the system’s speed to how quickly Twitter or crypto news sites pick up the same event (Tree, SolidIntel). If you’re consistently faster, you have an edge. If not, investigate latency bottlenecks.