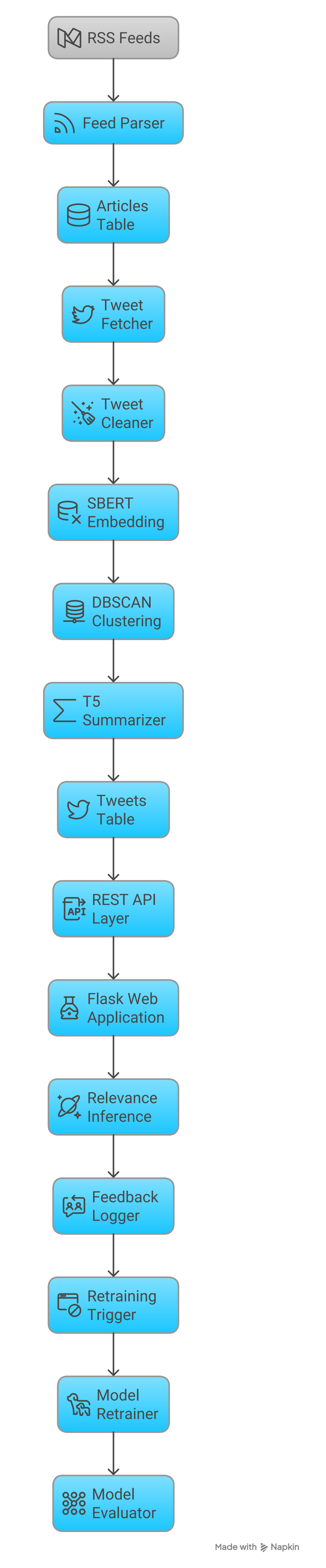
High -Level Data Diagram

Here is a comprehensive high-level diagram document for the Twitter News Sentiment Project:

Twitter News Sentiment Project Data Flow



The Project is split in two parts, First part, where we are fetching news, tweets, getting summary, sentiment and storing them in the database. In second part we are exposing the web interface to the user also using a model to show them relevant tweets and getting the feedback from the user for model feedback, and also dockerizing it.

📊 Server Side(News Fetching and summarization)

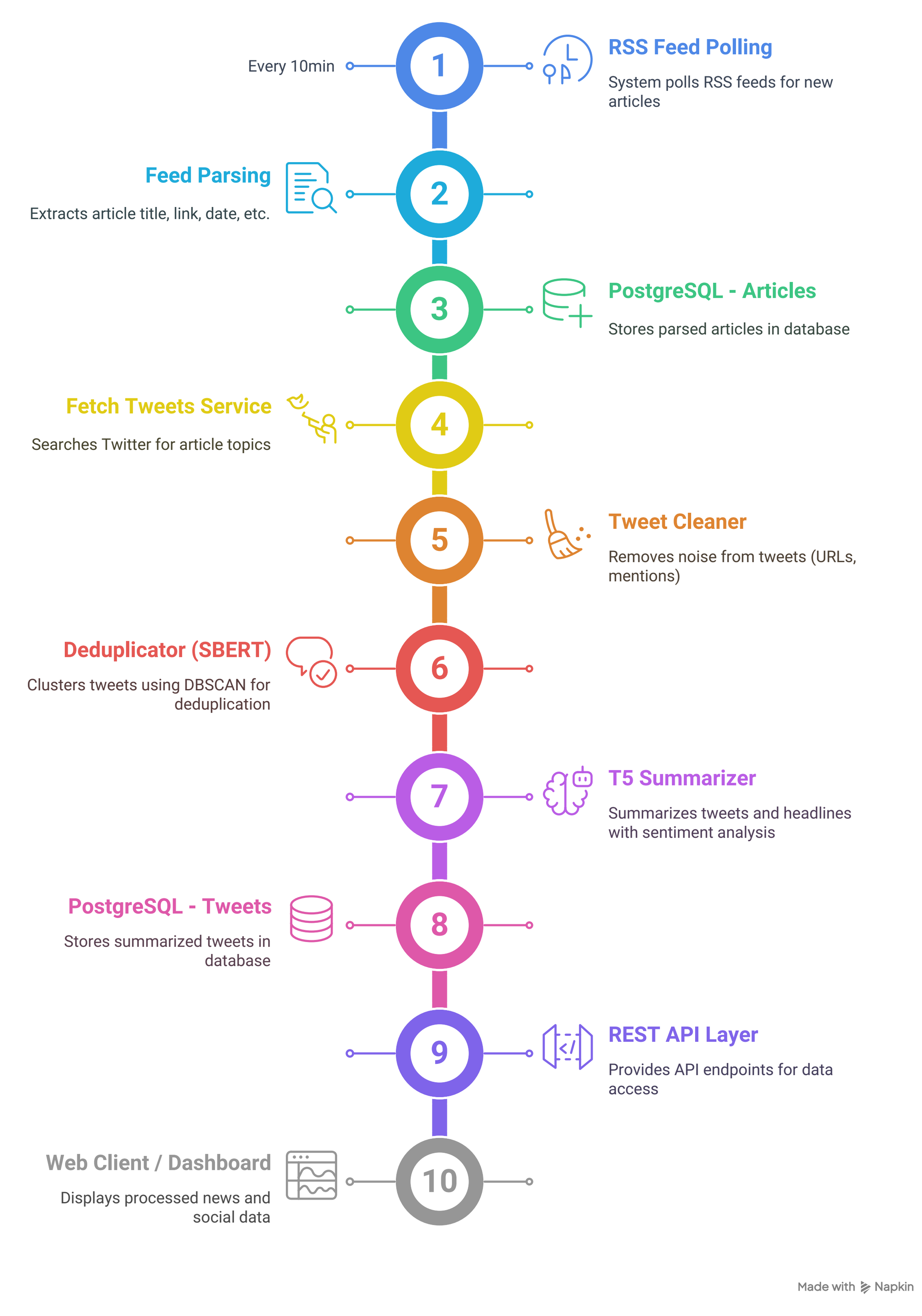
🧠 System Overview

The Twitter News Sentiment Project is a modular AI-driven pipeline that:

- **Aggregates news from RSS feeds**
- **Collects and filters tweets relevant to each article**
- **Applies NLP to deduplicate and summarize content**
- **Serves insights through a FastAPI-based RESTful API**

🔄 High-Level Data Flow Diagram

News and Social Media Data Processing Pipeline

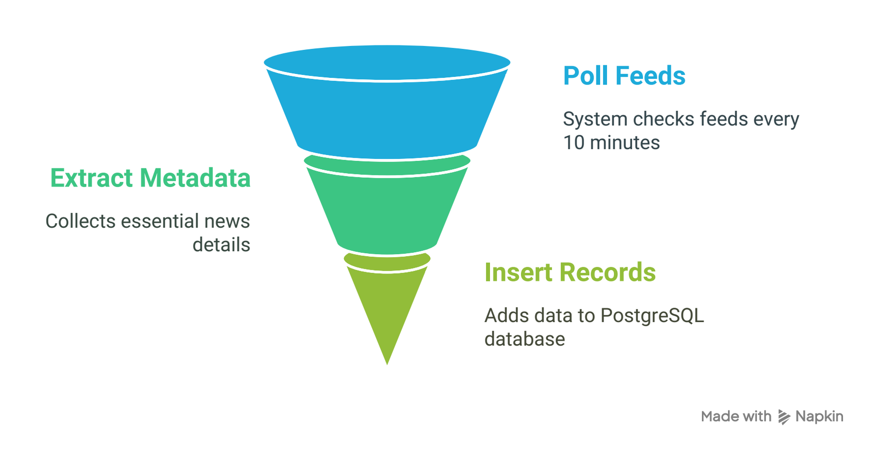


🔁 Detailed Data Flow

1. News Ingestion

- **Input: RSS feed URLs**
- **Process:**
  - **Feedparser polls feeds every 10 minutes**
  - **Extracts metadata: title, link, date, image, etc.**
- **Output: Records inserted into articles table in PostgreSQL**
- **Purpose: Establish baseline news entries for downstream tweet mapping**

News Ingestion Process



2. Tweet Collection & Cleaning

- **Input: News titles from the database**
- **Process:**
  - **fetch_tweets.py uses the Twikit client to query Twitter for each news title**
  - **clean_tweets.py strips tweets of URLs, hashtags, and mentions using regex**
- **Output: Cleaned tweets stored in memory for further processing**
- **Purpose: Gather relevant tweet activity and prepare it for semantic analysis**

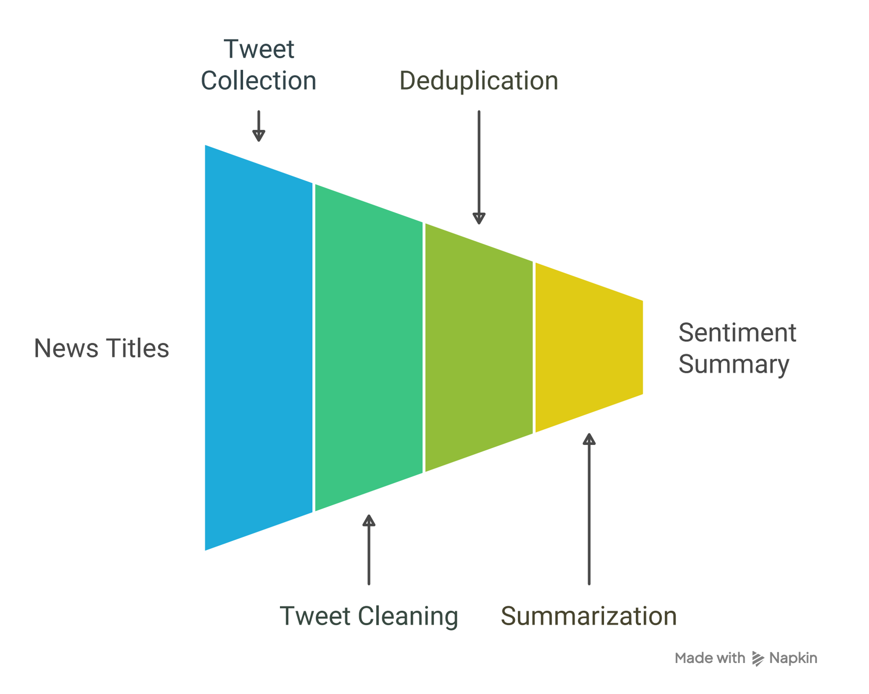
3. Deduplication

- **Input: Cleaned tweets**
- **Process:**
  - **Sentence-BERT generates embeddings**
  - **DBSCAN clusters semantically similar tweets**
  - **A representative tweet is selected per cluster**
- **Output: Clustered and representative tweets**
- **Purpose: Remove redundancy and improve summary quality**

4. Summarization (AI Analysis)

- **Input: Headline + Deduplicated tweets**
- **Process:**
  - **Run through a fine-tuned T5-base model**
- **Output: Text summary of tweet sentiment and themes**
- **Purpose: Generate high-level, human-like public reaction summaries**

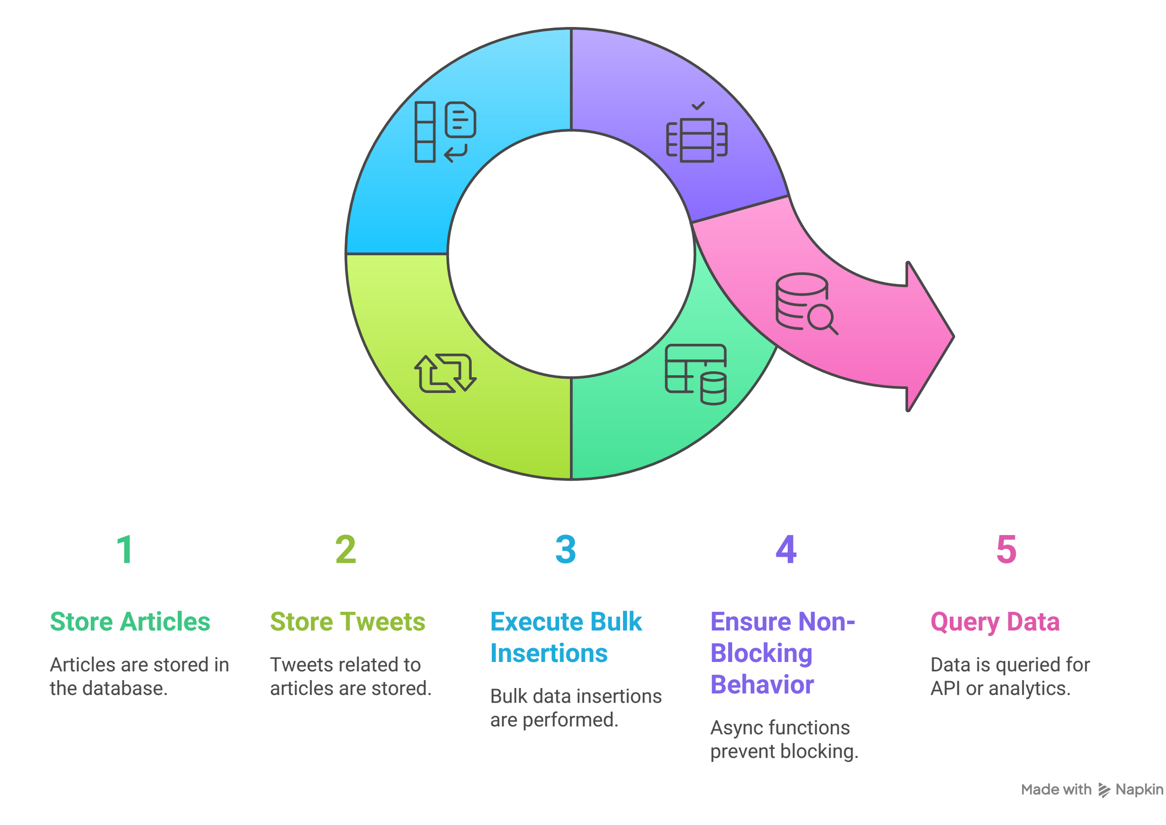
Tweet Analysis and Summarization Funnel



5. Database Storage

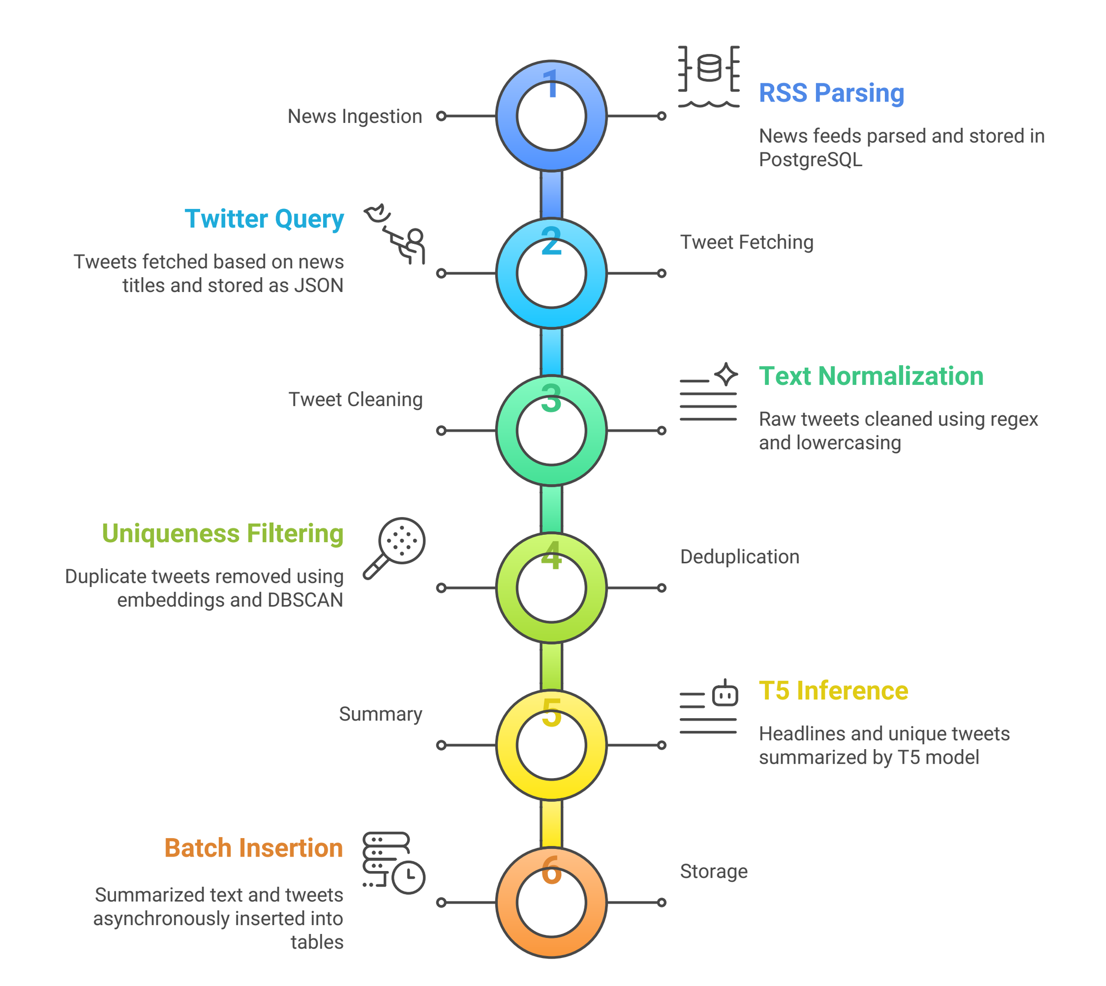
- **Tables:**
  - **articles: id, title, weblink, news summary, etc.**
  - **tweets: article_id, tweet_text, likes, retweets, etc.**
- **Process:**
  - **execute_values and execute_batch handle bulk insertions**
  - **Async functions ensure non-blocking behavior**
- **Purpose: Persistent storage for querying via the API or analytics layer**

Database Storage Cycle



📈 Data Transformation Stages

Data Transformation Pipeline Stages



📊 Web Interface(Feedback-Driven Learning)

The second part of the system extends the original machine learning pipeline into a **continuous learning architecture**, where the model improves over time based on real-time user input. The key idea is to **close the loop between model predictions, user judgment, and model retraining**, making the system adaptive and increasingly accurate in the wild.

🔁 Data Flow & System Functionality

User Interaction and Feedback Collection

The process begins on the **web interface**, where users browse through news articles and see tweets that have been automatically evaluated for relevance. Each tweet is accompanied by a prediction—"relevant" or "not relevant"—generated by the current machine learning model. Importantly, users are given the opportunity to **vote on whether they agree or disagree** with the prediction.
This interaction turns each user into a **real-time annotator**, supplying labeled data in a natural and frictionless way.

Feedback Accumulation and Logging

All user responses are captured and stored as structured feedback data. This includes the original inputs (headline and tweet), the model's prediction, and the user's correction. Over time, this log becomes a **growing dataset of real-world examples**, grounded in actual user perceptions of tweet relevance.
This is a crucial step that bridges the static nature of pre-trained models with the dynamic nature of real-world usage.

Trigger-Based Retraining

To avoid constant retraining and ensure efficiency, the system **monitors the volume of new feedback** and waits until a threshold (e.g., 50 new samples) has been reached. Once the threshold is met, the system sets a retraining flag. This mechanism decouples inference from learning, allowing the model to gather enough meaningful new data before updating itself.

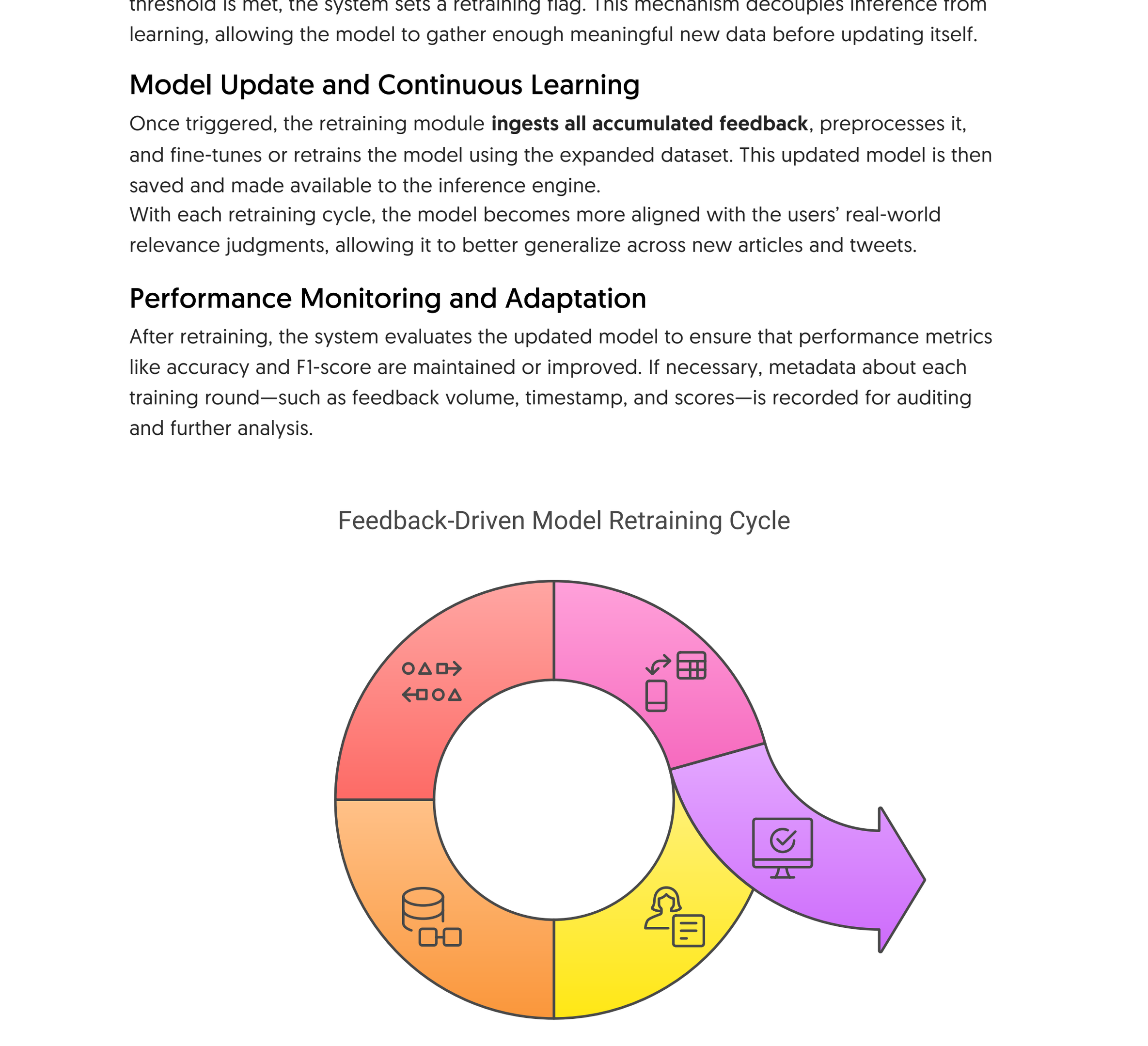
Model Update and Continuous Learning

Once triggered, the retraining module **ingests all accumulated feedback**, preprocesses it, and fine-tunes or retrains the model using the expanded dataset. This updated model is then saved and made available to the inference engine.
With each retraining cycle, the model becomes more aligned with the users' real-world relevance judgments, allowing it to better generalize across new articles and tweets.

Performance Monitoring and Adaptation

After retraining, the system evaluates the updated model to ensure that performance metrics like accuracy and F1-score are maintained or improved. If necessary, metadata about each training round—such as feedback volume, timestamp, and scores—is recorded for auditing and further analysis.

Feedback-Driven Model Retraining Cycle



This architecture enables a self-improving, user-aligned system. Rather than relying solely on static labeled datasets, it evolves continuously through real-world feedback—making the classifier more robust, adaptive, and trustworthy over time.