ChatGPT

Overview & Architecture

The **Institutional Rotation Detector** is built as a Node.js/TypeScript monorepo with Temporal workflows, Supabase (Postgres) for storage, and AI/ML components (GPT) [1] [2]. In this architecture, filings and positions are stored in a Supabase/PostgreSQL database (with pgvector for embeddings) [2], and Temporal workflows perform ingestion, analysis, graph building, etc. [1] [3]. To add the requested *real-time notification and tweet-generation* capability without hard-coding it into this project, we can design a **separate package/library** in the monorepo. This library would encapsulate the event-driven logic and social media integration, with clear TypeScript interfaces so it can be reused by other projects.

Event Detection (Database Triggers & Webhooks)

Since filings are already being inserted into the Supabase database, the new package can **listen for new rows on the filings table** and react. A robust way is to use **Supabase's database event triggers/ webhooks**. Supabase supports Postgres triggers (via the pg_net extension) that "send real-time data from your database to another system whenever a table event occurs" [4]. For example, you could configure a *Database Webhook* on the filings table for `INSERT` events. Each time a new filing row is added, Supabase can invoke a webhook (HTTP endpoint) asynchronously [4] [5]. The new package can provide that HTTP endpoint (e.g. a serverless function or a REST API route) to receive the webhook. Because Supabase's webhooks use pg_net under the hood, the main database insert isn't blocked by these callbacks [5].

Alternatively, if you prefer a poll-based or Temporal approach, you could schedule a Temporal workflow that queries the filings table for new entries and then emits events. But using Postgres triggers/ webhooks is more "real-time." In summary: **use a Postgres trigger or Supabase webhook on the filings table** so that each new filing insertion immediately triggers our package's logic [4].

New Package Design (Monorepo `libs/`)

In this pnpm monorepo, we'd create a new library under `libs/`, e.g. `libs/rotation-notifier` or `libs/filing-events`. This keeps code modular and shareable (see how an existing `openai-client` library is used for AI integration [6]). Inside this package, define clear **TypeScript interfaces** for event payloads and tweet content. For example, an interface `FilingEvent { ticker: string; filingType: string; data: any; }` can model incoming filing data. A separate interface for `TweetMessage { text: string; hashtags?: string[]; mediaUrls?: string[] }` can represent the crafted tweet. By codifying inputs/outputs, the package stays well-typed and reusable.

The new package would include code to: - **Register/listen** for filing events (e.g. an Express route or Supabase Edge function that the webhook calls). - **Process the event**: fetch any additional data needed (positions, past events, etc.), possibly via Supabase client (there's already a Supabase client in `apps/temporal-worker/src/lib/supabase.ts` or you can add one in this lib). - **Generate content**: e.g. call OpenAI (using the existing `openai-client` lib [6]) to summarize the event and create an engaging tweet. - **Post the tweet**: interface with Twitter's API or another social platform to publish the content (this could be another activity/function).

By packaging all this in `libs/rotation-notifier`, any other repo can reuse it by adding it as a workspace dependency. PNPM workspace hoisting and the `pnpm -r --filter './libs/**'` build scripts (shown in the root `package.json` [7]) will naturally build this lib along with the rest of the code.

Workflow/Activity Integration

Given the existing Temporal-based design [1] [8], one approach is to implement the notification as **a new Temporal workflow**. For example, a `filingNotifyWorkflow` that is started when a new filing arrives. The workflow can use `proxyActivities` to invoke activities like `composeTweetActivity` and `postTweetActivity`. The development guide even outlines how to add workflows and activities:

- *Add a new workflow file* `src/workflows/filingNotify.workflow.ts`, define its input interface, implement the logic, export it and write tests [9].
- *Add new activities* in `src/activities/` for tasks like content generation. For example, `composeTweet.activities.ts` could use the OpenAI client to produce a tweet text. The guide says to create or update an activity file and then use it via `proxyActivities` [10].

With Temporal orchestrating, the workflow could ensure retry/visibility etc. Alternatively, if you want to decouple from Temporal, the package could simply run as a standalone service (triggered by the webhook) and perform async tasks.

In either case, leverage the monorepo's structure: the `rotation-detector-temporal-worker` app can import the new lib to include these workflows, or the package can be used by any other Node/TS service. The **key is clear interfaces**: for example, define a TS type for the Supabase row that represents a filing (inserted from EDGAR) and use that in your logic. Then you can easily switch to a different source or project by implementing the same interface.

Generating "Viral but Valuable" Tweets

The tweet content should be both engaging and informative. We can use the project's existing AI infrastructure for this. For instance, the system already uses GPT (even GPT-5/CoT) for analysis and summaries [3]. We can repurpose that: in `composeTweetActivity`, call the OpenAI client (from `libs/openai-client`) with a prompt like:

> *"Given the following new institutional filing for {TICKER}, write a concise, insightful tweet (≤280 chars) that highlights why this is noteworthy. Include relevant hashtags (e.g. #stocks, #finance) and make it engaging."*

This uses Retrieval-Augmented Generation (RAG) style: you'd supply context (stock ticker, percent changes, institutions involved) and let GPT create human-like text. Ensure prompts steer GPT toward brevity and value (the dev guide suggests using "Chain of Thought" for complex summaries [3]).

Optionally, enrich the content using the knowledge graph/community data already in the system [1]. For example, you could query which funds were involved or related previous events, to add color to the tweet. But even without deep RAG, instructing GPT to write a tweet summary should yield quick results.

After generating the text, the package's `postTweetActivity` can call Twitter's API (using an npm Twitter client) to post. Make sure to handle API credentials securely (via environment vars, just like the existing OpenAI keys in `.env.local` [11]). The activities should be **idempotent** or properly rate-limited; for example, ensure you don't double-tweet the same event.

Putting It Together & Reusability

In practice, the flow would be:

1. **Trigger**: A new SEC filing row is inserted into Supabase.
2. **Webhook/Handler**: Supabase webhook calls our service endpoint (or triggers a Temporal workflow) with the filing info.
3. **Workflow Execution**: A workflow (or function) starts that does:
4. *Research step*: maybe analyze data or call an external service (e.g. check if the filing passes certain criteria).
5. *Tweet composition*: call `composeTweetActivity` to get tweet content.
6. *Posting*: call `postTweetActivity` to send it.
7. **Logging/Audit**: Record the tweeted event in the database (maybe an `events` table) so you avoid duplicates.

All code (workflows, activities, utilities) lives in the new `libs/` package with well-defined interfaces. For example, exports could include a `startFilingNotification` function and types `FilingEvent` and `TweetContent`. The host project (this repo or any other) would then import and wire up these pieces.

Finally, as the dev guide notes, any new code in this monorepo should come with tests [9]. You'd write unit tests for the activities (e.g. ensure `composeTweetActivity` returns a string with certain keywords), and workflow tests to simulate an end-to-end run. The existing test framework and scripts (`pnpm test`, Vitest, etc. in `apps/temporal-worker/__tests__`) can be used.

Summary: By creating a new TypeScript library in the monorepo (under `libs/`) that implements an event-driven architecture—using Supabase triggers or polling, Temporal workflows/activities, and the existing OpenAI integration—we can seamlessly add **filing-based notifications and tweet generation**. This package approach ensures clear interfaces and reusability. We leverage Temporal's orchestration and Supabase's real-time webhooks to detect new filings, then use GPT-based activities to craft tweets. The solution fits into the existing architecture of Temporal and Supabase [1] [4], while providing a modular, pluggable notification/tweeting system usable across projects.

Sources: The design follows the repository's existing architecture patterns [1] [3] and Supabase best-practices for database triggers/webhooks [4] [5], extended with a new workflow/activity as outlined in the project's development guide [9] [10].

---

[1] [2] [3] [6] [8] README.md
https://github.com/tradentic/institutional-rotation-detector/blob/023749dccc2f05e7124d8619f684fef03936d08c/README.md

[4] [5] Database Webhooks | Supabase Docs
https://supabase.com/docs/guides/database/webhooks

[7] package.json
https://github.com/tradentic/institutional-rotation-detector/blob/023749dccc2f05e7124d8619f684fef03936d08c/package.json

9 10 11 DEVELOPMENT.md
https://github.com/tradentic/institutional-rotation-detector/blob/023749dccc2f05e7124d8619f684fef03936d08c/docs/guides/DEVELOPMENT.md