**Explanations of Design Decisions**

**1. ChatGPT API for Summarization**

* **Why?** Human-readable summaries enhance the user experience by providing concise insights into large datasets.
* **How?** The application sends the top 10 posts of each day to the ChatGPT API, which generates a summary in natural language. These summaries are displayed on the /summary page.

**2. Server-Side Rendering**

* **Why?** Next.js 15’s App Router provides default SSR, ensuring faster page loads and better SEO.
* **How?** Data fetching is handled on the server, while interactivity like filtering is implemented on the client side.

**3. Dynamic Filtering with React Table**

* **Why?** @tanstack/react-table offers a customizable and performant table rendering solution.
* **How?** The table dynamically updates as the user adjusts the date range.

**4. Modular Components**

* **Why?** Reusability and scalability were key considerations during development.
* **How?** Components like DateFilter and Table were designed to be self-contained and reusable.

**Challenges Encountered**

**1. Integrating the ChatGPT API**

* **Challenge:** Sending large datasets to the ChatGPT API required careful prompt design to ensure meaningful responses.
* **Solution:** The prompt was structured to include only the most relevant data (e.g., top 10 posts with titles, scores, and authors) for each day.

**2. Combining SSR and Client-Side Interactivity**

* **Challenge:** Ensuring server-rendered pages also supported client-side interactivity for filtering.
* **Solution:** Separated server-side rendering for initial data fetching from client-side logic for filtering.

**3. Styling for Responsiveness**

* **Challenge:** Ensuring the table and date picker were fully responsive across devices.
* **Solution:** Used Tailwind CSS for consistent styling and responsive design.

**Project Structure**

graphql

CopyEdit

.

├── app/

│ ├── page.tsx # Home page with server-side data fetching

│ ├── summary/

│ │ └── page.tsx # Summary page with ChatGPT summaries

│ └── api/

│ └── getSummary/

│ └── route.ts # API route for generating summaries using ChatGPT API

├── components/

│ ├── Table.tsx # Table component using @tanstack/react-table

│ ├── DateFilter.tsx # Date range filtering and table integration

├── styles/ # Global and Tailwind CSS styles

└── README.md # Project documentation

**How Summaries Are Generated**

1. The /api/getSummary route fetches posts from the Supabase database.
2. Posts are grouped by date, and the top 10 posts for each day are selected based on their scores.
3. The top 10 posts are sent to the ChatGPT API with a structured prompt:

text

CopyEdit

Here is a list of the top 10 posts for [DATE]:

1. [Post Title] - Score: [Score], Author: [Author]

2. ...

Please summarize these posts in a concise and engaging manner.

1. The API returns a natural language summary, which is displayed on the /summary page.

**Future Improvements**

* Add pagination for large datasets in the table.
* Enhance the summarization API to support multi-language summaries.
* Implement authentication for restricted access to certain pages.
* Optimize API calls with caching for better performance.

**License**

This project is licensed under the MIT License.

**Contact**

For any questions or suggestions, feel free to reach out:

* Email: tri09t4@gmail.com
* GitHub: <https://github.com/AI-LastWish/TrendHive>