State Affairs: Take-Home Assignment: Full Stack Engineer Role

Legislative News Aggregator

Objective:

Create a **Legislative News Aggregator** where users can view the latest news articles related to state legislations and filter the news by specific states or legislative topics (e.g., education, health, transportation). This assignment will test your ability to build a system that pulls, stores, and displays news from external sources, while also focusing on UI, API integration, and efficient data storage.

Requirements:

1. **Functionality:**
    - **View News Articles**: Display a list of news articles related to state legislation. Each article should include:
        - Title
        - Published date
        - State
        - Summary or description
        - Link to the full article
    - **Filter News**: Allow users to filter news by state (e.g., California, Indiana) and/or by topic (e.g., education, health).
    - **Search Articles**: Implement a simple search functionality to find news articles by keywords in the title or description.
    - **News Aggregation**: Simulate fetching news articles from an external API or a local dataset. You can use dummy data or an actual news API (like the NewsAPI or any other Public News API from this list).
2. **Back-End:**
    - Use **Node.js** with **Express** (or **Next.js API routes**).
    - Implement the following API endpoints:
        - **GET /news?state=x&topic=y&search=keyword**: Retrieve a list of news articles filtered by state, topic, or search keywords.
        - **GET /news/**
          : Retrieve detailed information about a specific news article.
        - **POST /news** (optional): Simulate the ability to add new articles to the system (useful for testing how new data is integrated).
    - **Data Storage**: Use an in-memory store like **SQLite**, **Redis**, or a simple database (**MySQL**, **PostgreSQL**) to store news articles with relevant metadata.
    - Use **TypeScript** for type safety and clean code.

    - For **news aggregation**, simulate the process of pulling news from multiple sources (you can create a script that runs periodically or allow for manual article addition via an API).
3. **Front-End:**
    - Use **React** with **Next.js** (or a standalone React app).
    - Create the following UI components:
        - **News List**: Display the list of news articles with basic metadata (title, summary, state, topic, date).
        - **Filter & Search**: Implement a filtering UI where users can filter articles by state, topic, and perform keyword searches.
        - **News Details**: When clicking on an article, show more detailed information about the article, along with a link to the full text.
        - **Responsive Design**: Ensure the UI is functional and responsive on both desktop and mobile.
4. **System Design Considerations: [**via README file**]**
    - **News Aggregation**: Describe in your README how you would aggregate news articles from multiple sources. How would you handle deduplication, store articles, and ensure fresh data?
    - **Scalability**: Discuss how the system could handle thousands of news articles across multiple states and topics. Would you index the articles? What storage strategies would you use?
    - **Search Optimization**: Consider how you would make searching through potentially large datasets efficient.
5. **Performance and Caching:**
    - Implement basic caching for news articles to avoid repeatedly fetching or processing the same data, especially if simulating API calls or fetching from an external source.
    - Discuss or implement pagination for the news list if there are many articles.
6. **Security Considerations (Optional):**
    - Ensure that user input for filtering and searching is properly sanitized to avoid injection attacks.
    - (Optional) Implement rate-limiting on the news fetching endpoints to prevent abuse.
7. **Bonus Points (Optional Features):**
    - **Real-time Updates**: Simulate real-time updates to the news list to notify users when new articles are published.
    - **User Personalization**: Implement a way for users to save their preferred states and topics, so they automatically see relevant articles when they return to the site.
    - **Newsletter Subscription**: Allow users to subscribe to a daily email summary of the latest news articles related to their saved states/topics.

Submission:

- Submit the project via **GitHub**, with a clear **README** that includes setup instructions, a description of the system design, and notes on scalability and performance considerations.
- Please make assumptions, if something is not clear and communicate those assumptions clearly in README or code comments.
- Include any advanced features or extra details on how the system could be improved.
