**INSIGHTSTREAM**

**Introduction**

* + Project Title: Insightstream.
  + Project ID: SWTID1741244875153895
  + Team Leader: SANTHOSH KUMAR T
  + Team Member: SIVAMANI P
  + Team Member: DHEENATHAYALAN P
  + Team Member: SANJAY K
  + Team Member: DANIEL B

**Project Overview**

**Purpose:**

Provide users with curated news, articles, or blog posts.Organize content into categories like General, Technology, Science, Lifestyle, Health, Fitness, etc.Enable easy navigation and user engagement with content.

**Features:**

1. **Home Page:Displays trending articles (TopStories.jsx).**

**Highlights featured content.**

1. **Category-Based Navigation:Users can browse articles by categories (Footer.jsx links). Uses React Router for seamless navigation.**

**3. Authentication & User Management (if implemented):Users might be able to sign up, log in, and manage preferences.Clerk authentication could be integrated if user profiles are required.**

**4. Newsletter Subscription:Component (NewsLetter.jsx) suggests a subscription option.Users may receive updates on trending topics.**

**5. Article Management:Components like HomeArticles.jsx suggest content fetching/display.Admins or authors may upload articles via a CMS-like interface.**

**6. Responsive UI & Styling:Uses CSS Modules or global styles (Footer.css).Likely optimized for both desktop and mobile views.**

**Architecture**

**Component Structure:**

* + - React Import: Required for JSX rendering.
    - CSS Import: Styles are applied from an external stylesheet (Footer.css).
    - useNavigate Import: A hook from react-router-dom for handling navigation.
    - Functional Component: Defined using an arrow function.
    - Navigation Hook: useNavigate is used to programmatically navigate between pages.
    - Navbar.jsx - Manages the navigation bar
    - Footer Wrapper (<footer>): Encapsulates the entire component.
    - Main Content (<div className='footer'>): Contains the navigation list.
    - Navigation List (<ul>):Each <li> has an onClick event that triggers navigate() to the respective category page.

**State Management:**

* Uses React's useState and useEffect hooks for local state management
* Data fetched and managed using axios
* useContext for global state (optional for future expansion)

**Routing:**

* Utilizes react-router-dom
* Routes include:
  + - * / - App.js
      * / package.json
      * /Category.jsx
      * /Footer.jsx

**Setup Instructions**

**Prerequistes:**

* + - Install [Node.js](https://nodejs.org/en/download/" \t "_new)
    - Install Git for version control ([Git Download](https://git-scm.com/downloads" \t "_new))
    - IDE: [VS Code](https://code.visualstudio.com/download" \t "_new)

**Installation:**

* git clone [repository\_url]
* cd Recipies-app
* npm install

**Start the application:**

* cd path/to/your/recipe-app
* npm install
* npm start
* npm run start

**Folder Structure**

stream-app/

│── node\_modules/

│── public/

│ ├── favicon.ico

│ ├── index.html

│ ├── logo192.png

│ ├── logo512.png

│ ├── manifest.json

│ ├── robots.txt

│── src/

│ ├── components/

│ │ ├── Footer.jsx

│ │ ├── Hero.jsx

│ │ ├── HomeArticles.jsx

│ │ ├── NavbarComponent.jsx

│ │ ├── NewsLetter.jsx

│ │ ├── TopStories.jsx

│ ├── pages/ (This folder is visible but contents are not shown)

│ ├── styles/

│ ├── App.css

│ ├── App.js

│ ├── App.test.js

│ ├── index.css

│ ├── index.js

│ ├── logo.svg

│ ├── reportWebVitals.js

│ ├── setupTests.js

│── gitignore

│── package.json

│── README.md

**Running the Application**

* **Frontend:** npm run dev

**Component Documentation**

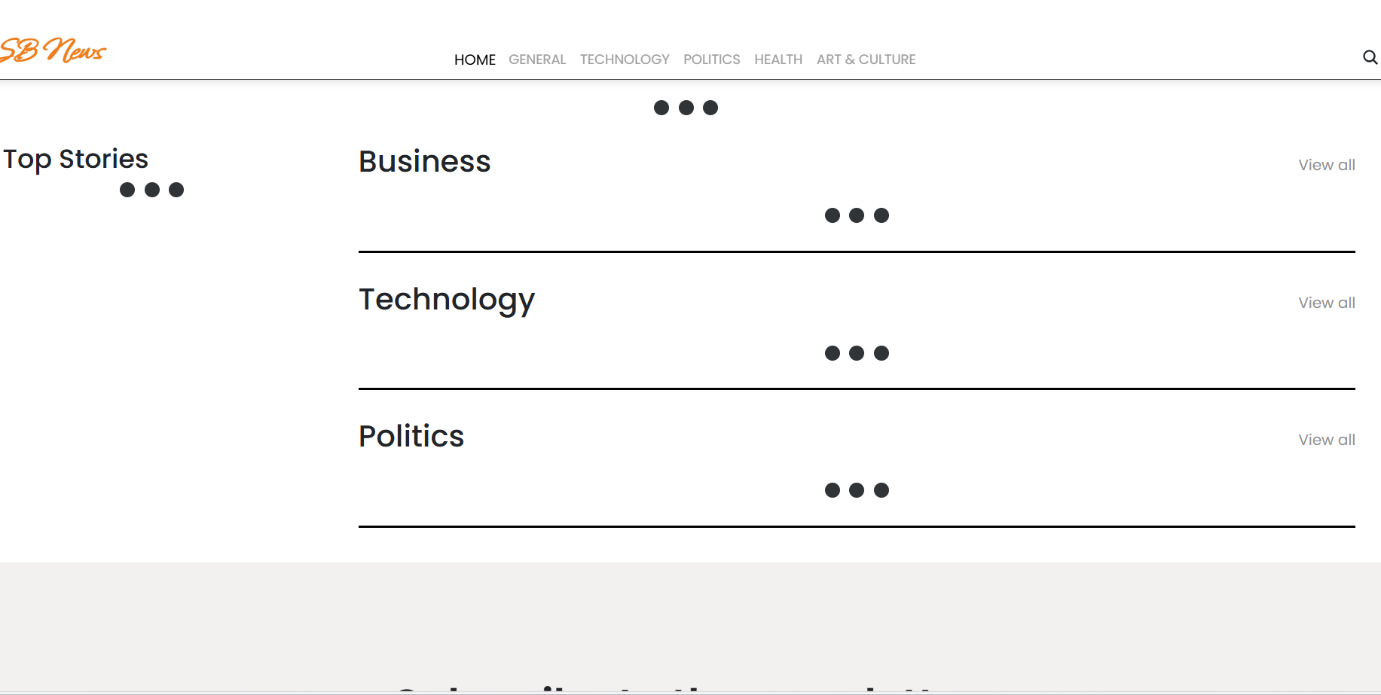
**Key Components:**

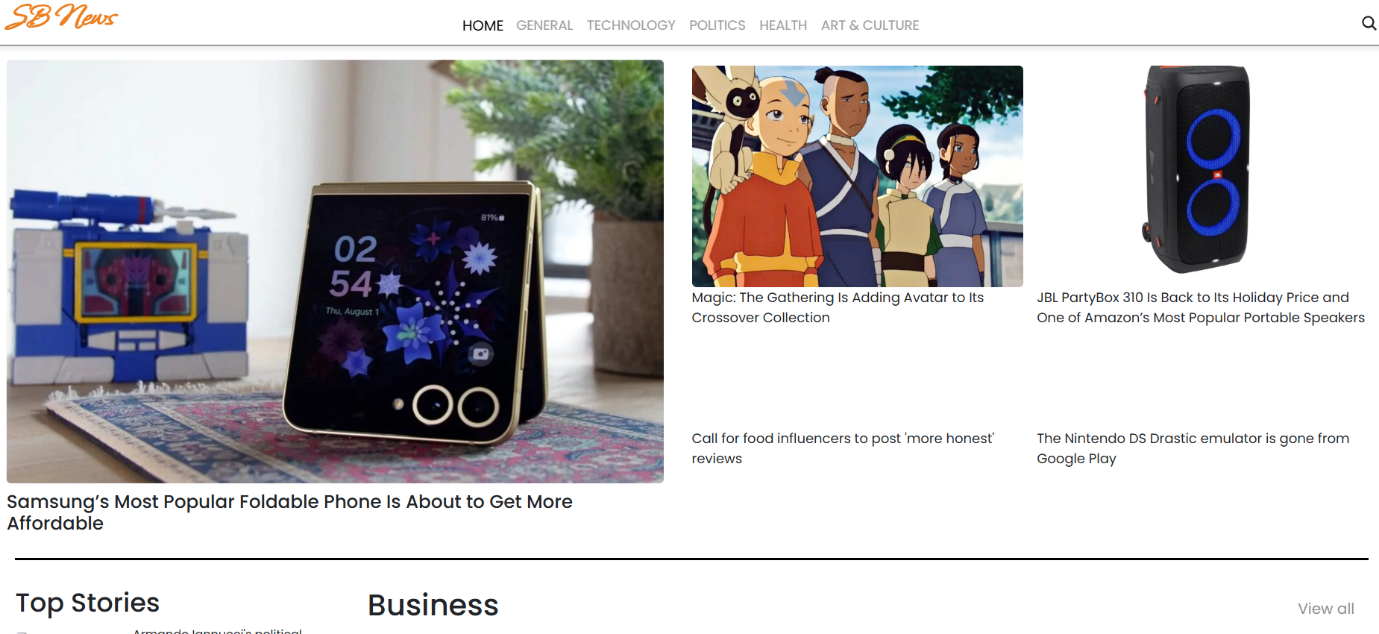
* + - index.js : The main entry point of the React app.
    - App.js → Handles routing between different pages.
    - pages/Home.js → Displays featured categories and recent recipes.
    - pages/homearticle.js → Fetches and displays a list of recipes

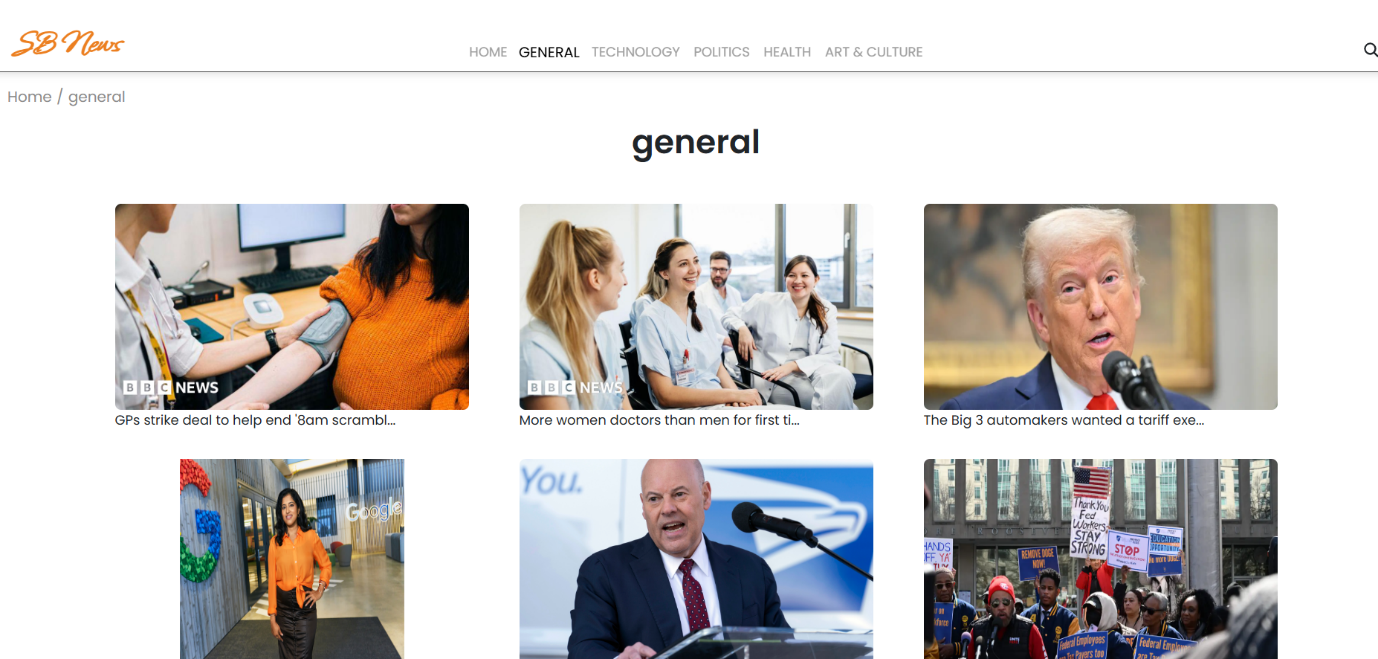
**State Management**

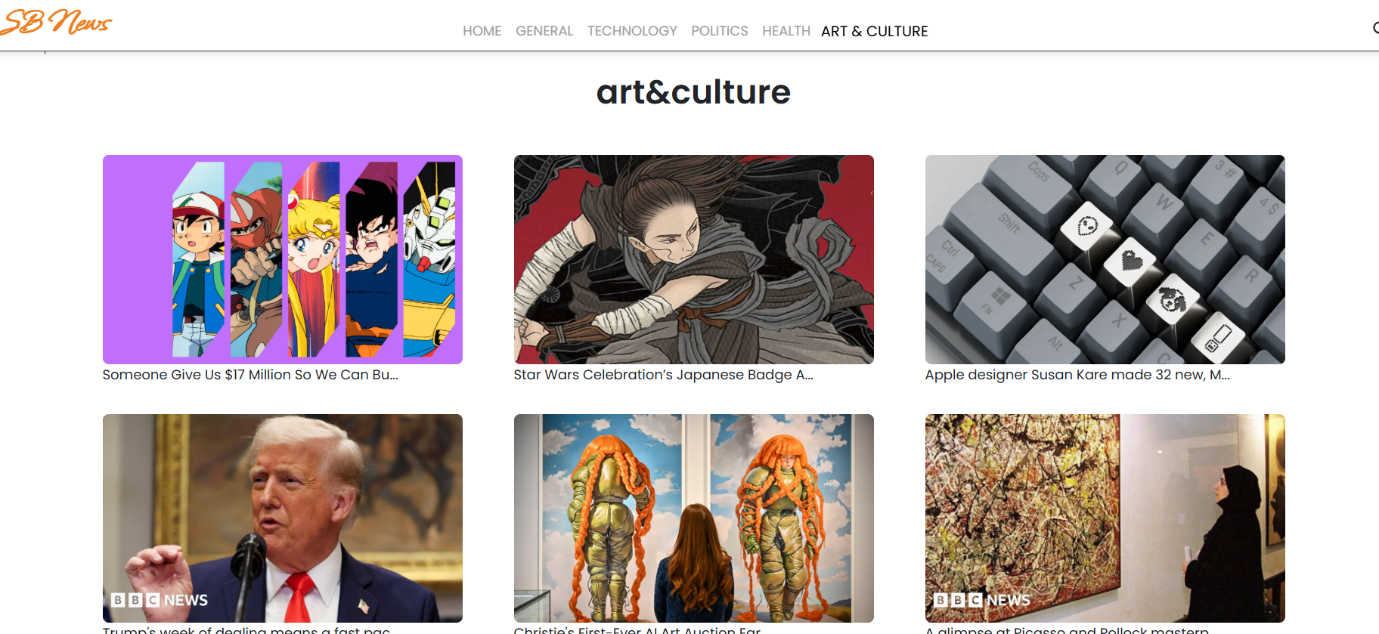
* **Global State:** Not yet implemented (potentially using Context API)
* **Local State:** Managed with useState and API calls with axios

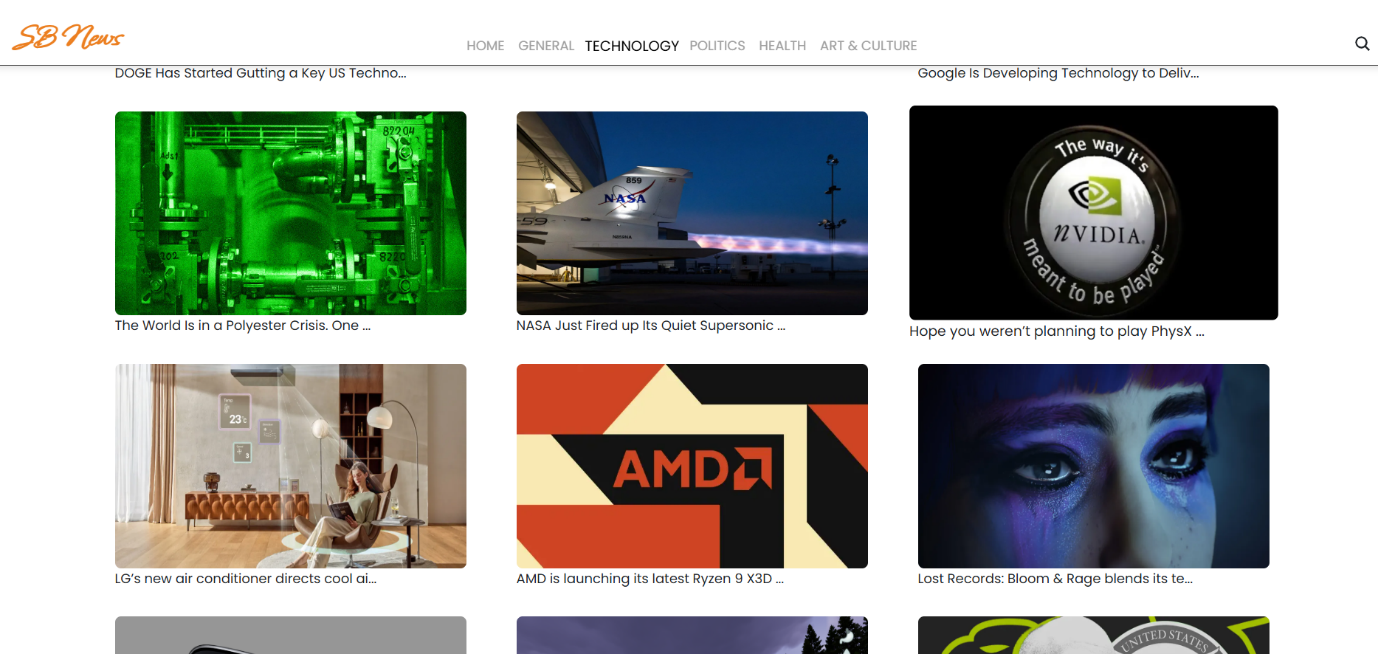
**User Interface**

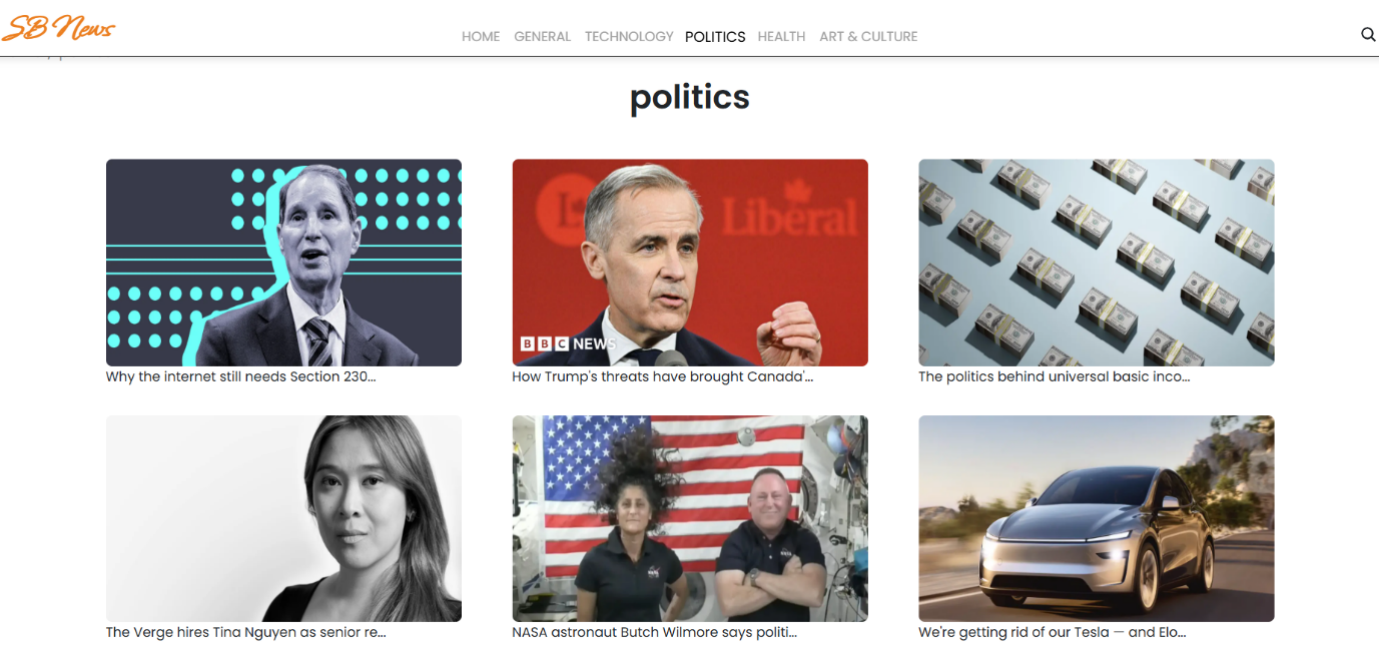
****

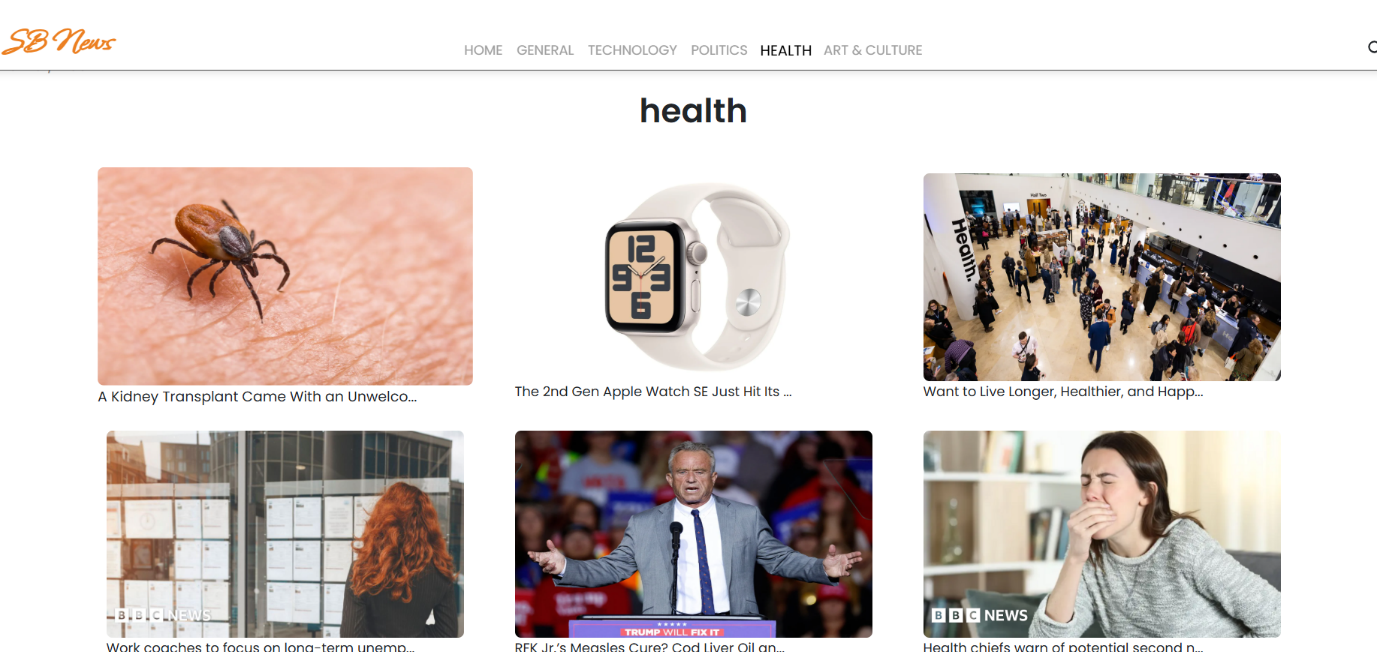
****

****

****

****

****

****

**Styling**

**CSS Frameworks/Libraries:**

* + - Bootstrap/Tailwind CSS
    - Custom CSS for additional styling

**Testing**

**Testing Strategy:**

* Unit tests with Jest & React Testing Library (future enhancement)

**Code Coverage:**

* No test coverage tool implemented yet

**Demo:**

https://drive.google.com/drive/folders/1yyyuel4fNBoQBcNwkIyl4SRIrPpBy1S9?usp=sharing

**Known Issues**

* Common challenges in stream processing include scalability issues, inadequate resource allocation, bottlenecks in data processing, and limitations in handling peak loads.
* Solutions involve horizontal scaling and utilizing specialized stream processing platforms like Apache Kafka or Apache Flink

**Future Enhancements**

* Allow users to switch between light and dark mode.
* Add an option for custom themes.
* Implement a global search bar with autocomplete suggestions.
* Add filters by date, category, popularity, etc.
* Implement a "For You" section with personalized recommendations based on reading history.
* Allow users to save/bookmark articles for later.