PROJECT DOCUMENTATION

Title: Inside stream-navigate-the-news-landscape

Team Project ID : NM2025TMID37517

Team Size : 4

Team Leader:

Name : Dharshini.T

Email id : dharshinitamilselvam382

Team Members:

1. **Name :** Amsa.V

   Email id : aamsa472@gmail.com

2. **Name :** Anju.V

   Email id : venkateshsandhiyaanju@gmail.com

3. **Name** :Ganga Bhavani L

   Email id : gangabhavaniganga83@gmail.com

2. INTRODUCTION :

In today's digital era, news plays a vital role in shaping public opinion and keeping people informed about the world around them. With the rise of online platforms, social media, and digital journalism, the news landscape has become vast, fast, and sometimes confusing. People often face challenges in identifying trustworthy sources, filtering relevant updates, and avoiding misinformation.

Our project, "Inside Stream: Navigate the News Landscape", aims to provide a clear path for users to explore, analyze, and understand news effectively. By studying the news ecosystem, we highlight how technology, media channels, and user behavior influence the way information is delivered and consumed.

3. Project Overview :

Purpose :

The purpose of this project is to simplify news consumption for readers, ensure access to verified information, and create an easy way for people to explore trending topics, categories, and regions of interest.

Problem Statement :

Today's readers face challenges such as information overload, fake news, biased reporting, and difficulty in finding news that matches their interests. These issues reduce trust in media and make it harder for people to stay updated with genuine news.

Solution :

Our project offers a centralized platform that aggregates news from reliable source uses AI to filter out misinformation, and allows user to customize their news feed. With features like category sorting, keyword search, fact-check indicator, and summarized highlights, inside stream ensure users can navigate the news landscape quickly and confidently.

Key Benefits :

• Provides verified and reliable news.

• Saves time by filtering unwanted or irrelevant content.

• Reduce the risk of misinformation.

• Offers personalized and Category - wise news suggestions.

• Enhances awareness of current events with simplified summaries.

Features :

• Personalized news feed based on user interests.

• Fact- checking integration to highlight authentic news.

• Category filters (politics, sports, technology, health, etc.).

• News summarization for quick reading.

• Multi- source comparison to reduce bias.

• Search and trending topics for easy navigation.

4. Architecture :

Inside Stream: Navigate the News Landscape project, designed for efficient news fetching, storage, and presentation:

1.Frontend (client side) :

Technology:

HTML / CSS / JavaScript

Framework: React.js / Angular / Vue.js (depending on implementation)

Functionality:

Displays news articles in a clean, user-friendly interface.

Dynamic Search Bar to allow keyword-based article search.

Category Filters for browsing different news types (e.g., Technology, Sports, Business).

Interactive UI Components for user engagement.

Workflow:

1.User interaction:

User searches for a keyword or selects a news category in the web app.

2.API Request:

Frontend sends a request to the backend to get relevant articles.

3.Data fetching:

Backend checks the database.

If data is outdated or missing → Fetches fresh articles from third-party News APIs.

4.Data response:

Backend sends the article data (title, description, URL, image) as JSON to the frontend.

5.Display and update:

Frontend displays articles to the user, and periodic background tasks keep the database updated automatically.

Prerequisites :

Install Node.js and npm for running JavaScript and managing packages.

Install React.js to build the frontend UI:

➔ **Create app using:**

Npx create-react-app insightstream-app

Code Editor: Visual Studio Code recommended

Set up a Database (MySQL, PostgreSQL, or MongoDB) to store fetched news articles.

For deployment (optional):

Frontend → Netlify / Vercel

Backend → Heroku / AWS / DigitalOcean

Steps (Frontend Only):

1.Install Node.js and npm

2.Create a new React project:

Npx create-react-app insightstream-app

3.Navigate to project folder:

Cd insightstream-app

4.Install Axios for API requests:

Npm install axios

5.Build React components:

Header

SearchBar

CategoryList

NewsCard

Footer

6.Integrate News API to fetch articles using Axios and your API key.

7.Display fetched articles in a list or grid on the UI.

8.Start the development server:

Npm start

9.Open in browser:

http://localhost:3000

10.Apply CSS or use Tailwind CSS / Bootstrap for styling.

Steps (Full-Stack Version) :

1. Clone repository:

git clone <repo-link>

2. Install client dependencies:

.cd client

. Npm install

3. Install server dependencies:

cd ../server

npm install

4. Start frontend:

Start frontend

1. Navigate to the frontend project folder:

Cd insightstream-app

2.Start the development server:

Npm start

5. Start backend: npm run server

6. Folder Structure:

Simple frontend Version

```
  Insightstream-app/        # Root folder (React project)
|
 ├── public/              # Public assets (favicon, index.html, images)
|   └── index.html
|
 ├── src/                 # Main source code
|    ├── components/       # Reusable UI components
|   |    ├── Header.js
|   |    ├── SearchBar.js
|   |    ├── CategoryList.js
|   |    ├── NewsCard.js
|   |    └── Footer.js

|    ├── pages/            # Page-level components
|   |    └── Home.js
```

```
|   ├── services/         # API calls
|   |   └── newsApi.js

|   ├── App.js            # Root component
|   ├── index.js          # Entry point
|   └── App.css           # Global styles
|
 ├── package.json          # Dependencies & scripts
 ├── package-lock.json
 └── README.md
```

Explanation :

1.Public/ → Contains static files like index.html, favicon, and images that do not change.

2.Src/ → Main source code of the project.

Components/ → Reusable UI parts (Header, SearchBar, CategoryList, NewsCard, Footer).

Pages/ → Page-level components (e.g., Home.js for main page).

Services/ → API call files (e.g., newsApi.js to fetch news).

App.js → Root component that connects all parts.

Index.js → Entry point of the React app.

App.css → Global styles for the app.

3.Package.json → Contains project details, dependencies, and scripts (npm start, npm install, etc.).

4.README.md → Project documentation and setup instructions.

Full-Stack Version :

Insight stream – App/

```
Insightstream-project/          # Root folder
|
├── client/                  # Frontend (React.js)
|   ├── public/              # Static files (index.html, favicon, images)
|   ├── src/                 # React source code
|   |   ├── components/      # Reusable UI components
|   |   |   ├── Header.js
|   |   |   ├── SearchBar.js
|   |   |   ├── CategoryList.js
|   |   |   ├── NewsCard.js
|   |   |   └── Footer.js
|   |   ├── pages/           # Page-level components
|   |   |   └── Home.js
|   |   ├── services/        # API calls
|   |   |   └── newsApi.js
|   |   ├── App.js           # Root React component
|   |   ├── index.js         # Entry point for React
|   |   └── App.css          # Global styles
|   ├── package.json         # Frontend dependencies
|   └── README.md
|
├── server/                  # Backend (Node.js / Express)
|   ├── config/              # Configuration files
|   |   └── db.js            # Database connection
|   ├── models/              # Database schemas
|   |   └── Article.js
|   ├── routes/              # API routes
|   |   └── newsRoutes.js
|   ├── controllers/         # Business logic for APIs
```

```
|   |   └── newsController.js
|   ├── server.js             # Entry point for Express server
|   ├── package.json          # Backend dependencies
|   └── .env                  # Environment variables (API keys, DB URI, PORT)
|
├── node_modules/             # Installed packages
|
└── README.md                 # Full project documentation
```

7. Running the Application :

The backend server is started first — it connects to the database, fetches news data from APIs, and provides it to the frontend.

The frontend server is then started — it runs the user interface in the browser where you can search, filter, and read the news.

React frontend :

Go to the client folder → cd client

Start the React frontend →npm start

Backend:

Go to the server folder → cd server

Start the backend server → npm start

8. API Documentation (Optional Backend) :

GET /api/news → Fetch all news articles

GET /api/news/:id → Get details of a specific news article

GET /api/categories → Fetch all news categories

POST /api/news → Add a new news article

DELETE /api/news/:id → Delete a news article

9. User Interface :

Home Page: Shows featured news articles with images and headlines.

News List Page: Displays news articles in card format (title + image + category + published date).

News Detail Page: Shows full details of a news article (title, author, content, source, published date).

Search & Filter Page: Allows searching articles by keyword and filtering by category.

Responsive Design: Works smoothly on desktop, tablet, and mobile devices.

10. Testing :

Manual Testing:

Checked on Chrome, Firefox, and Edge browsers.

Test Case:

Fetch news articles → Articles appear correctly on the homepage and news list.

Search by keyword → Displays relevant news articles matching the query.

Filter by category → Shows only articles of the selected category.

View article details → Displays full content, source, author, and published date.

Responsive Design → Layout works correctly on desktop, tablet, and mobile.

Tools Used:

Browser DevTools: For inspecting elements, checking console errors, and testing responsiveness.

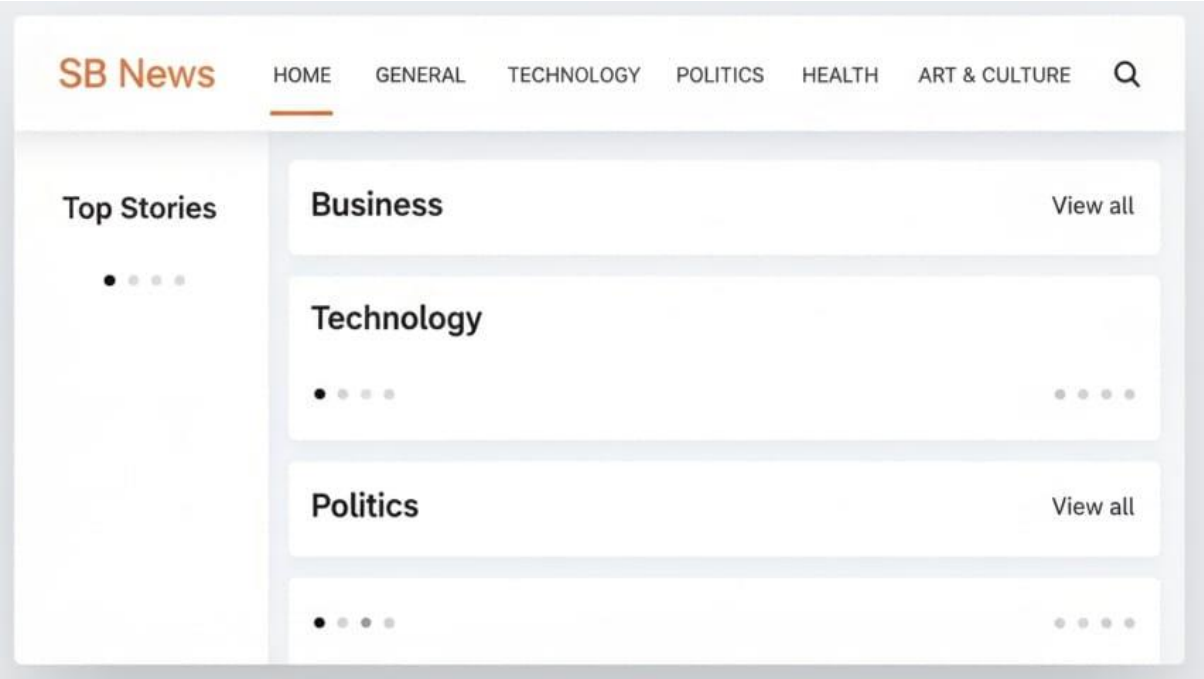
Postman: For testing backend APIs (optional, if APIs are used).

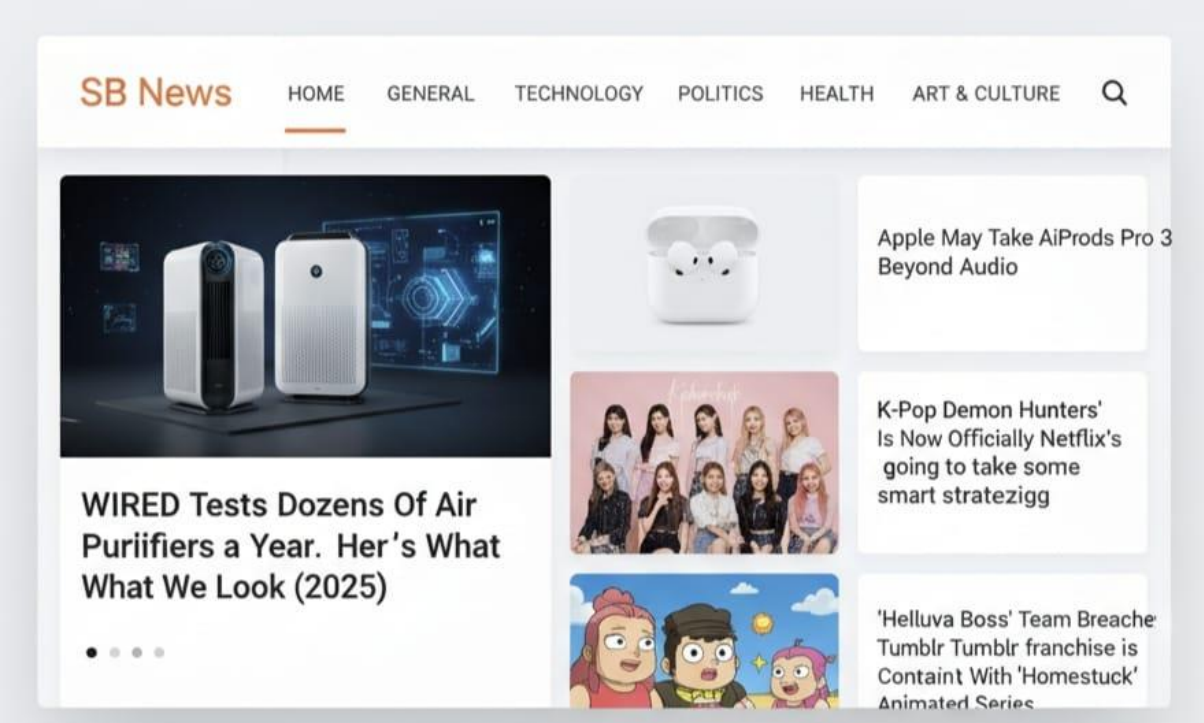
Visual Studio Code: Code editor for frontend and backend development.

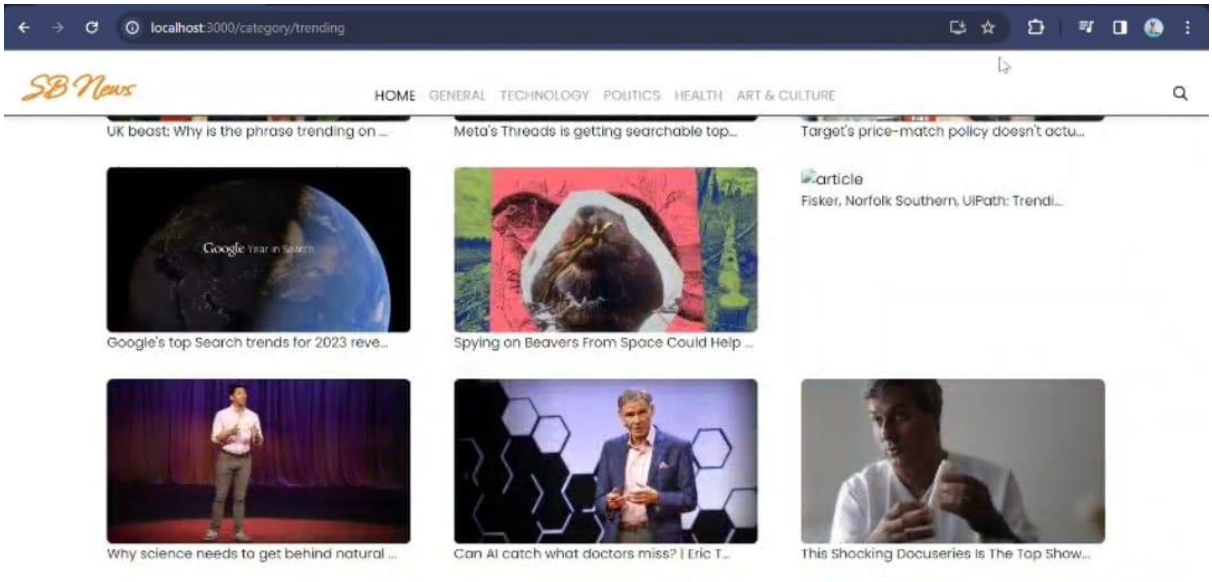
Git & GitHub: Version control and repository management.

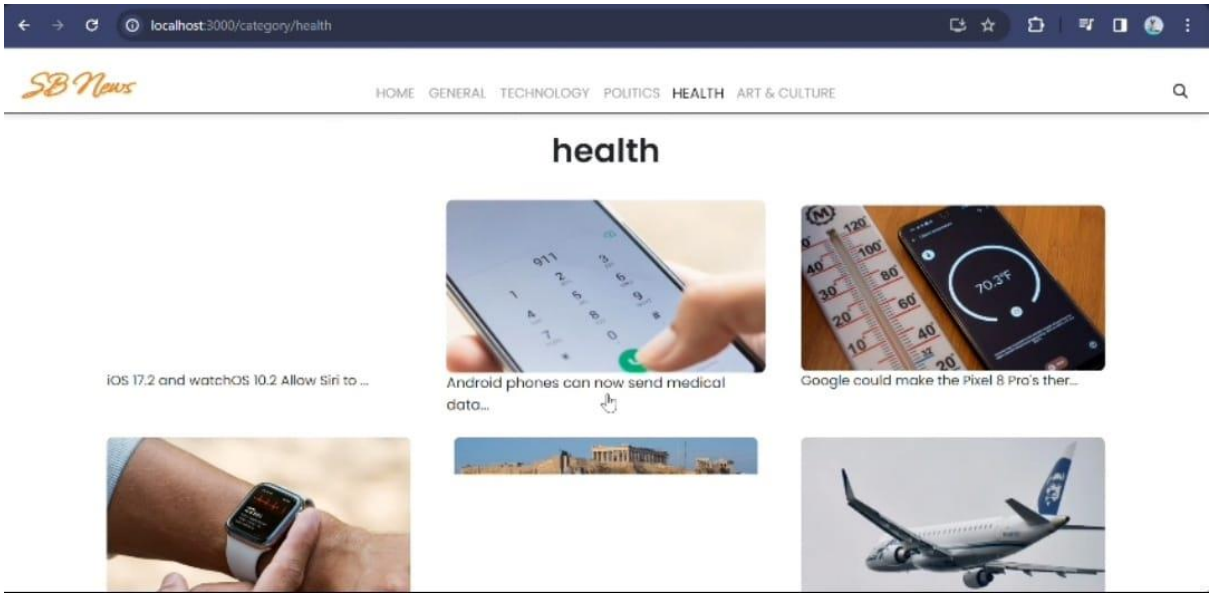
Node.js & npm: For running the backend and managing project dependencies.

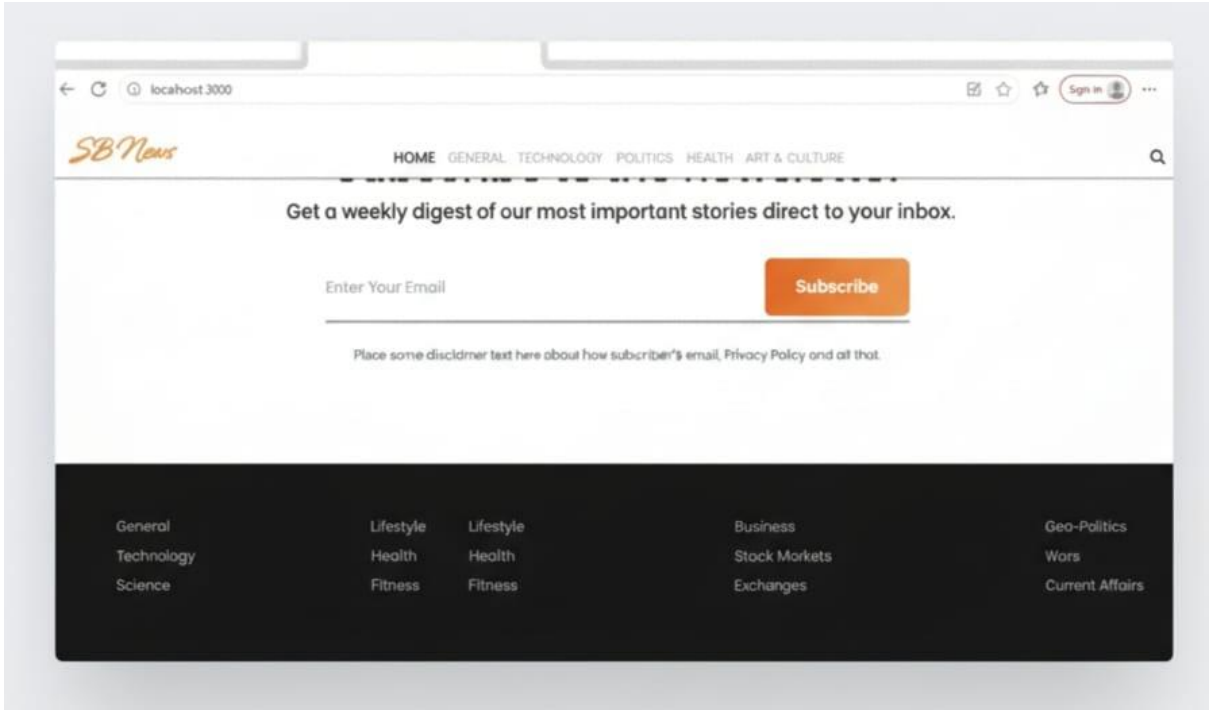
React Developer Tools: For debugging React components.




SB News
HOME
GENERAL
TECHNOLOGY
POLITICS
HEALTH
ART & CULTURE
WIRED Tests Dozens Of Air Puriifiers a Year. Her's What What We Look (2025)
Apple May Take AiProds Pro 3 Beyond Audio
K-Pop Demon Hunters' Is Now Officially Netflix's going to take some smart stratezigg
'Helluva Boss' Team Breache Tumblr Tumblr franchise is Containt With 'Homestuck' Animated Series


localhost:3000/category/trending
SB News
HOME
GENERAL
TECHNOLOGY
POLITICS
HEALTH
ART & CULTURE
UK beast: Why is the phrase trending on ...
Meta's Threads is getting searchable top...
Target's price-match policy doesn't actu...
Google's top Search trends for 2023 reve...
Spying on Beavers From Space Could Help ...
article
Fisker, Norfolk Southern, UiPath: Trendi...
Why science needs to get behind natural ...
Can AI catch what doctors miss? | Eric T...
This Shocking Docuseries Is The Top Show...





12. Known Issues:

Some news articles may not load properly if the external source is unavailable.

User preferences are not saved in the frontend-only version (lost after refresh).

Search results may sometimes show irrelevant articles due to limited filtering.

No login system in the basic version, so personalized features are unavailable.

13. Future Enhancements:

Add login/signup with JWT authentication for personalized experience.

Save user preferences and bookmarked articles permanently in a database (MongoDB/MySQL).

Add categories and filters (Politics, Technology, Sports, Entertainment, etc.).

Implement a rating and comment system for articles.

AI-based news recommendations based on user interests and reading history.

Real-time notifications for trending news or topics.