**PROJECT NAME :INSIGHTSTREAM:**

**NAVIGATE THE NEWS LANDSCAPE**

**(NEWS APP)**

**TEAM MEMBER AND THEIR ROLES:**

K.SUGANYA(TEAM LEADER AND TESTING)

S.VAISHNAVI(FRONTEND DEVELOPER)

B.SUBHASHINI(BACKEND DEVELOPER)

S.SYEDA HAJIRA PARVES(EDITING)

**PROJECT OVERVIEW:**

**Purpose:** INSIGHTSTREAM aims to revolutionize the way users consume news by providing a personalized, intuitive, and comprehensive news application. Its primary goal is to deliver tailored news content, ensuring users stay informed about topics that matter most to them.

**Features:**

**Personalized News Feed:** Utilizes advanced algorithms to curate news based on user preferences, reading habits, and interests, ensuring relevant content delivery.

**Real-Time Updates:** Provides timely news alerts and updates, keeping users informed about the latest developments as they happen.

**Multimedia Integration:** Incorporates articles, videos, podcasts, and infographics, offering diverse content formats to cater to various user preferences.

**User Engagement Tools:** Features options for users to comment, share, and discuss articles, fostering a community-driven news experience.

**Offline Access:** Allows users to download articles and read them offline, ensuring uninterrupted access to news content.

**Cross-Platform Synchronization:** Ensures seamless access across devices, with user preferences and reading history synchronized for a consistent experience.

**Advanced Search Functionality:** Enables users to search for news articles using keywords, topics, or dates, facilitating easy discovery of specific content.

**Customizable Notifications**:Allows users to set preferences for news alerts, ensuring they receive notifications about topics of interest without being overwhelmed.

**Trusted Sources:** Aggregates news from reputable sources, ensuring accuracy and reliability of the information presented.

**User Privacy and Security:** Prioritizes user data protection with robust security measures and transparent privacy policies.

**ARCHITECTURE:**

**1. Component Structure:**

The app follows a modular component structure where each major feature and view of the app is encapsulated into individual React components. Below is a basic outline of the component structure and how they interact:

**App Component:**

The entry point of the application where all major routing and global contexts are set up.

Includes a wrapper for routing and the main layout.

**Main Layout (Layout Component):**

Contains the overall structure of the application, including the header, sidebar, footer, and main content areas.

Acts as a container for all major sections of the app (e.g., navigation, news articles, trending topics).

**Header Component:**

Displays the navigation bar with links to the homepage, search, and category sections (e.g., Politics, Sports, Technology).

Includes search functionality and user authentication options (login/logout).

**News Feed Component:**

Displays a list or grid of news articles retrieved from an API.

Each article is rendered as a card or item that contains the article's title, image, summary, and a link to the full article.

Interacts with the API to fetch the latest news based on selected category or search queries.

**Article Page Component:**

Displays detailed information for a single article when clicked from the news feed.

This page may include the full article text, related news links, and options to share or save the article.

**Category Filters Component:**

A sidebar or dropdown that allows users to filter news by categories like sports, politics, entertainment, etc.

This component interacts with the news feed component to update the displayed articles based on the selected category.

**2. State Management:**

The state management approach used in INSIGHTSTREAM can either use Context API for simple state sharing or Redux for more complex scenarios.

**Context API:**

The app uses Context API to manage global states such as user authentication, current news category, and search query across different components.

**Redux (optional for complex state):**

If the app requires more advanced state management (e.g., managing large data sets, handling async requests, etc.), Redux can be introduced for global state management.

**Redux can be used to manage:**

**News Articles Data:** Fetching and caching articles.

**User Data:** Managing user preferences and personalized settings.

**UI State:** Managing modal visibility, loading states, etc.

Action creators and reducers will be defined for different actions like fetching news articles, updating categories, and handling search results.

**3. Routing:**

The routing structure of INSIGHTSTREAM is handled using react-router for dynamic routing between different pages in the app.

**SETUP INSTRUCTION:**

**Prerequisites**

Before you can start setting up make sure you have the following software dependencies installed:

**1. Node.js**

Node.js is required for running JavaScript on the server side and managing the project's package dependencies.

**2. npm** (Node Package Manager)

npm comes bundled with Node.js, but ensure it is updated by running:

npm install

**3. Git** (for cloning the repository)

If Git is not already installed, you can download it from here.

**INSTALLATION:**

**1. Clone the Repository**

First, clone the repository to your local machine using Git. Open your terminal or command prompt and run the following commands.

**2. Install Dependencies**

Run the following command to install all required dependencies specified in the package.json file:

npm install

This will install all the libraries and packages required to run the app.

1. **Start the Development Server:**

Once the dependencies are installed and the environment variables are set, you can start the development server to run the app locally:

npm start

This will launch the app in your default web browser at [**http://localhost:3000.**](http://localhost:3000.)

**FOLDER STRUCTURE:**

Explanation of Major Folders and Files

**1. public:**

Contains the index.html file that serves as the template for the React app.

favicon.ico for the site’s favicon.

**2. src**:

assets: Holds images, icons, and other static files required by the application.

For example, images contains images for the news articles and icons holds smaller UI icons used in the app (e.g., search icon, category icons).

**Components:** Contains reusable UI components that can be shared across multiple pages or sections of the app.

Components like Header.js, Footer.js, NewsCard.js, SearchBar.js, and CategoryFilter.js are present here.

**pages:** Contains components representing full pages or views of the app.

Each page is a separate React component (e.g., Home.js, Category.js, SearchResults.js, ArticlePage.js), designed to represent a distinct route in the application.

**context:** Contains React context files for global state management, ensuring certain pieces of state are accessible across the app.

**For example:**

**AuthContext.js:** Manages authentication state (logged in user, token, etc.)

**NewsCategoryContext.js:** Holds the current selected news category for filtering.

**SearchQueryContext.js:** Stores the current search query for filtering articles based on user input.

**useSearch.js:** Custom hook for managing the logic of searching and filtering articles.

services: Contains files that manage API calls and data fetching.

**newsApi.js:** Handles requests to external APIs for news data.

**authApi.js:** Deals with user authentication via API (e.g., login, registration).

**categoryApi.js:** Handles fetching of news articles based on categories.

**utils:** Contains utility functions that help with common tasks throughout the app.

**formatDate.js:** A helper function to format the date of news articles.

**debounce.js:** A utility to limit the number of times a function (e.g., search query) is invoked within a short time period.

**constants.js:** Stores constants like API endpoints or fixed values used in multiple parts of the app.

**App.js:** The root component that initializes and renders the app.

**index.js:** The entry point to the application, where the App component is rendered and linked to the root DOM element.

**RUNNING THE APPLICATION:**

**1. Navigate to the Project Directory:**

Make sure you are in the root folder of the project. If you're not already there, navigate to it using the terminal:

cd /path/NEWS APP

**2. Install Dependencies**

If you haven’t already installed the necessary dependencies, run the following command to install all required packages:

**npm install**

This will ensure all dependencies in the package.json file are installed.

**3. Start the Frontend Development Server**

Once the dependencies are installed, you can start the local development server by running:

**npm start**

This command will start the React development server and automatically open the app in your default web browser. The app will be available at **http://localhost:3000.**

**4. Access the Application**

After running the above command, the React app should be accessible at:

**http://localhost:3000**

You can now interact with the app, test out features, and start developing locally.

Stopping the Server

**COMPONENT DOCUMENTATION:**

**1. App.js**

Purpose:

The root component of the application that renders the entire app and manages routing, global contexts, and layout.

Props:

None directly; this component wraps around all routes and global contexts (e.g., authentication, search query).

**2. Header.js**

Purpose:

Displays the top navigation bar, which includes links to home, search, and category pages. It also contains the search bar for filtering articles.

Props:

onSearchChange (Function): A callback function for handling changes to the search query (from the SearchBar).

**3. Footer.js**

Purpose:

Renders the footer at the bottom of the page with additional links like "About Us," "Contact," and "Privacy Policy."

Props:

None; it is a static footer.

**4. NewsCard.js**

Purpose:

A reusable component that represents a single news article card. It displays the article’s title, image, description, and a link to the full article.

Props:

**title (String):** The title of the article.

**description (String):** A brief description or summary of the article.

**imageUrl (String):** URL for the image or thumbnail associated with the article.

**articleUrl (String):** The URL of the full article (link to the external news source).

**5. CategoryFilter.js**

Purpose:

Allows users to filter news articles by specific categories (e.g., Politics, Sports, Technology). It can be used as a sidebar or dropdown.

Props:

**categories (Array):** An array of category names that the user can select from (e.g., ['Politics', 'Sports', 'Technology']).

**onCategoryChange (Function):** A callback function to handle category changes (updates the context or parent component).

**6. NewsFeed.js**

Purpose:

Displays a list of NewsCard components, which represent individual news articles. It fetches and filters articles based on the current category or search query.

Props:

**articles (Array):** Array of article objects containing the data to be displayed (e.g., title, description, image, URL).

**isLoading (Boolean):** A flag indicating whether the articles are still being loaded.

**error (Boolean):** A flag indicating whether there was an error fetching the articles.

**7. ArticlePage.js**

Purpose:

Displays the full content of a single article when the user clicks on a NewsCard.

Props:

**articleId (String):** The unique ID of the article, used to fetch its details.

**articleData (Object):** Contains the full data of the selected article (e.g., title, content, author, publication date)

**8. SearchBar.js**

Purpose:

A search input field that allows users to search for specific news articles.

Props:

**query (String):** The current search query entered by the user.

**onSearchChange (Function):** A callback function that updates the query state as the user types.

**STATE MANAGEMENT:**

State management plays a crucial role in React applications, as it ensures that data flows smoothly between components, allowing the app to be dynamic and interactive. In INSIGHTSTREAM, we use both global state and local state to manage the app's data and interactions.

**Global State Management**

The global state refers to data that needs to be accessible across various components throughout the app, and is shared by multiple components. This is particularly useful for things like authentication status, user preferences, or filtering criteria that should persist across different views or pages.

**1. Context Providers**

We use multiple context providers to manage the state for different concerns:

**AuthContext:** Handles the authentication state, such as whether the user is logged in and user information.

**NewsCategoryContext**: Manages the current category of news being viewed (e.g., Technology, Sports, Politics).

**SearchQueryContext:** Keeps track of the user's search query for filtering articles.

These context providers are initialized in the App.js file, wrapping the whole app with the appropriate context providers so that any child component can access the data and dispatch actions as necessary.

**Local State Management**

Local state refers to state that is specific to a component and does not need to be shared with other parts of the application. It is typically used for managing internal UI state, such as form inputs, toggles, or temporary values.

**1. SearchBar.js:**

The SearchBar component uses local state to manage the input value as the user types a search query.

Local state explanation :

The query state is l SearchBar component, and it changes as the user types in the input field. Once the input changes, the handleChange function updates the local state and then calls the onSearchChange function, which can update the global state or trigger a search action in the parent component.

**2. NewsFeed.js:**

The NewsFeed component uses local state to manage loading and error states during the news fetching process.

**USER INTERFACE:**

**1. Home Page (News Feed)**

**Features:**

Displays a list of news articles as NewsCard components.

Each NewsCard shows the article title, image, and brief description.

Clicking on a NewsCard takes the user to the full article page.

**Interactions:**

Scroll through the list of articles.

Click on an article to view more details.

Screenshot/GIF Example: Capture the home page with a scrollable list of news cards. Show a hover effect or click interaction on one of the cards, transitioning to the article details.

**2. Search Bar Interaction**

**Features:**

The search bar at the top allows users to enter a query to filter the news feed.

Real-time updates of news articles as the user types in the search query.

**Interactions:**

Type a search query in the SearchBar.

The results update dynamically to reflect the search.

Screenshot/GIF Example: Show a user typing in the search bar, with the news feed updating as they type. You could also show no results found if there is no match.

**3. Category Filter Sidebar**

**Features:**

The user can filter news articles by category (e.g., Politics, Technology, Sports).

The articles update based on the selected category.

**Interactions:**

Click to select a category (e.g., "Technology").

Articles in the feed update based on the selected category.

Screenshot/GIF Example: Show the category sidebar with a user clicking on a category (e.g., "Technology") and the articles changing accordingly.

**4. Article Page**

**Features:**

A detailed page for a single article, showing the full content, author, publication date, and related articles or recommendations.

**Interactions:**

Scroll through the article's content.

Click links to related articles.

Screenshot/GIF Example: Capture the detailed view of an article page with a user scrolling through the content. Highlight elements like the article title, author name, and full text.

**5. Authentication (Login/Logout)**

**Features:**

A login form with fields for username/email and password.

Once logged in, the user’s name is displayed in the header, with a logout button.

**Interactions:**

Fill in the login form and submit.

Log out by clicking the logout button.

Screenshot/GIF Example: Show the login screen with the user entering credentials, followed by the transition to the homepage after successful login. Optionally, show the logout process.

**STYLING:**

CSS Frameworks/Libraries Used

**Tailwind CSS**

Description:

Tailwind CSS is a utility-first CSS framework used in INSIGHTSTREAM to speed up development and provide a consistent design. It allows developers to apply utility classes directly in HTML (or JSX) to style elements without writing custom CSS rules for each element.

**Theming and Custom Design System**

**1. Theming with Tailwind CSS**

Description:

INSIGHTSTREAM leverages Tailwind CSS for custom theming through its configuration file (tailwind.config.js). The configuration allows defining custom colors, fonts, spacing, breakpoints, and other utility classes that can be reused across the app.

Tailwind’s default theme can be customized, and custom themes are defined in this configuration to align with the app’s branding and design guidelines.

**TESTING:**

Testing is an essential part of building a reliable and maintainable application. In INSIGHTSTREAM, a combination of unit tests, integration tests, and end-to-end (E2E) tests ensures that the app's functionality is correct, the components interact as expected, and the overall user experience is seamless. Below is an overview of the testing strategy used in the app.

**Testing Strategy**

The testing approach for INSIGHTSTREAM follows a layered strategy involving unit, integration, and end-to-end (E2E) testing. This strategy helps ensure that each part of the app is thoroughly tested at different levels, ensuring both correctness and reliability.

**1. Unit Testing**

Unit tests are used to test individual functions or components in isolation. The goal is to verify that each component works as expected in isolation, without dependencies or integrations with other parts of the application.

Tools Used:

**Jest**: The primary testing framework used in the app for running unit tests.

**React Testing Library:** A library designed to test React components by rendering them and simulating user interactions.

**Example Unit Test**: Let's say you have a simple SearchBar component. A unit test might check if the input field correctly updates the state when a user types.

**2. Integration Testing**

Integration tests verify that different parts of the app work together as expected. For instance, an integration test might ensure that the NewsFeed component correctly fetches data from an API and renders it in a NewsCard component.

**Tools Used:**

**Jest:** For running the tests.

**React Testing Library:** To render components and interact with them.

**Mock Service Worker (MSW):** For mocking API responses during testing.

**3. End-to-End (E2E) Testing**

End-to-end tests are used to simulate real user interactions within the application, ensuring that the app functions as expected from start to finish. These tests verify that the app performs as intended in a real-world scenario, including routing, user authentication, data fetching, and UI updates.

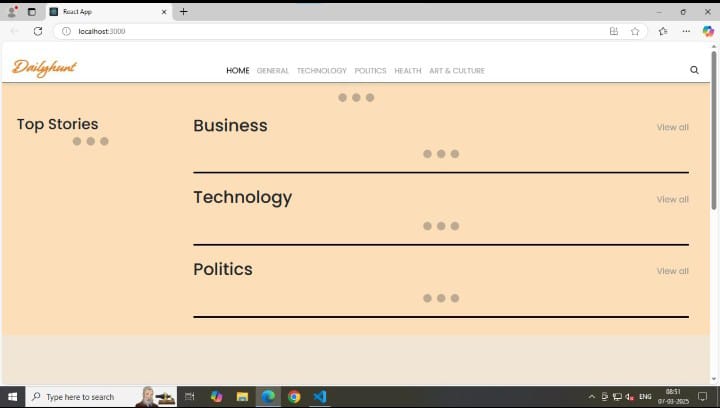
**Tools Used:**

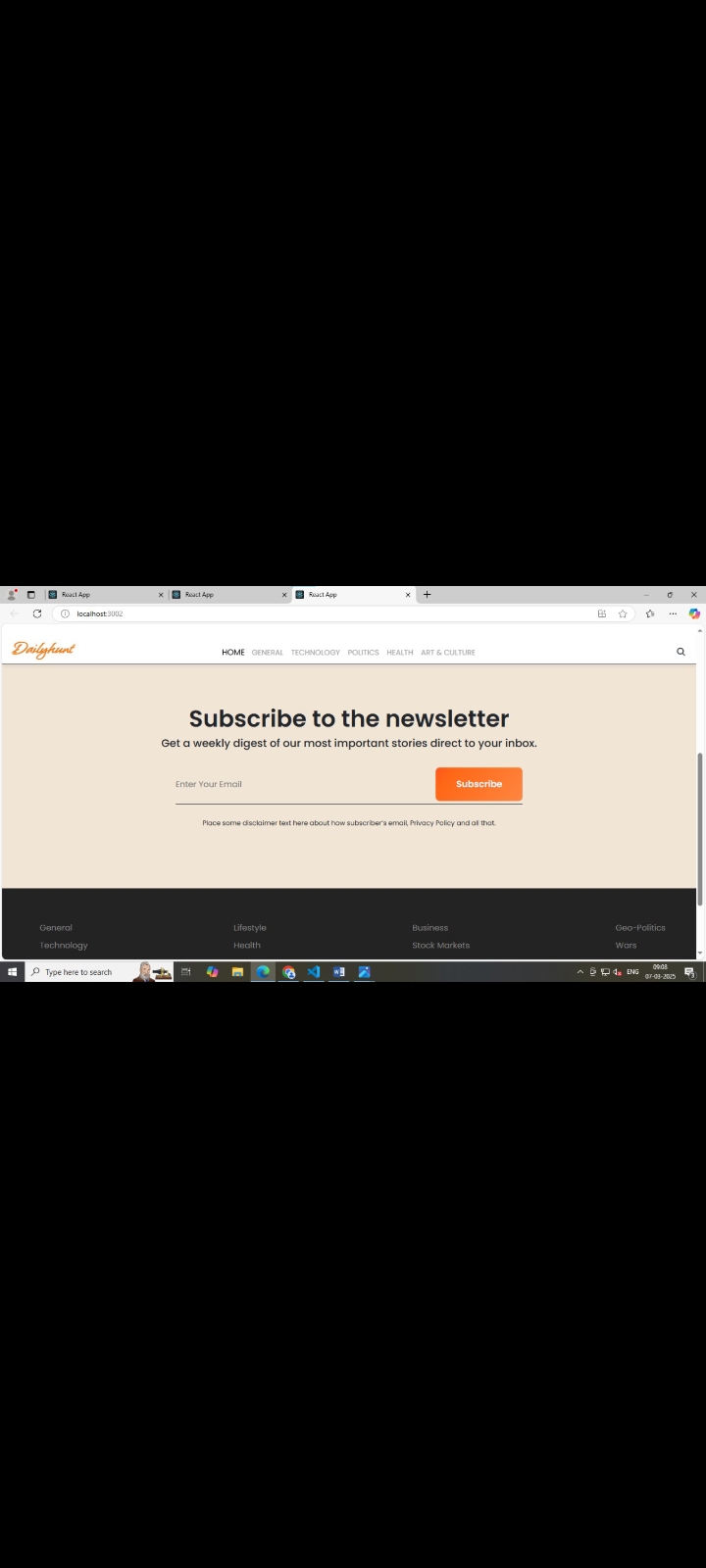
Cypress: A powerful framework for writing and running E2E tests.

**Jest:** For integrating tests and providing an overall test suite.

**Example E2E Test:** This test might simulate a user visiting the homepage, performing a search, and logging in.

**SCREENSHOTS OR DEMO:**





**KNOWN ISSUES:**

**1. Search Functionality Delays**

**Issue:** The search results may not update immediately when typing long queries or when the user enters characters too quickly. This issue is related to debounce handling, which may delay the API call.

**Expected Behavior:** The search results should update dynamically as the user types.

**Current Workaround:** There is no official workaround at this time. Users are advised to wait for the search results to load or avoid typing very quickly.

**2. Broken Links to External News Articles**

**Issue:** Occasionally, certain external links in the NewsCard components may be broken or lead to a 404 page when clicked. This happens when the external news source changes its URL or removes the article.

**Expected Behavior:** Clicking on a NewsCard should always lead to a valid article page.

**Current Workaround**: If a user encounters this issue, they can either refresh the page or manually visit the homepage and choose another article to read.

**Planned Fix:** We plan to introduce link validation during the fetching process to handle and flag broken links before they are presented to users.

**3. User Authentication (Login) Session Expiry**

**Issue:** Users who are logged in may experience unexpected session expiry, requiring them to log in again after a certain period. This is due to the current session management mechanism, which doesn’t handle token expiration well on the frontend.

**Expected Behavior:** Users should remain logged in until they explicitly log out or their session reaches its expiration.

**Current Workaround:** Users can manually log back in if they are logged out unexpectedly.

**Planned Fix:** The session management system will be updated to handle token expiration more gracefully and provide users with an automatic re-authentication flow.

**FUTURE ENHANCEMENT:**

**1.Offline Mode:** Implement an offline mode where users can view previously loaded content even without an internet connection, improving usability in low-connectivity areas.

**2. Search Functionality:** Add a search bar to allow users to easily find specific news articles or topics within the app, making it more user-friendly and efficient.

**3. Article Bookmarking:** Enable users to bookmark or save articles for later reading. This can help users keep track of important stories they want to revisit.

**4. News Personalization:** Introduce personalized news feeds based on user preferences, history, or reading habits, ensuring the content is relevant to each individual user.

**5. Voice Search Integration:** Implement voice search functionality, allowing users to search for news or perform actions using voice commands, adding convenience and accessibility.

**6.Content Filtering:** Allow users to filter news articles based on categories, sources, or dates, enabling a more tailored and focused reading experience.

**THANK YOU**