### **InsightStream:Frontend Development with React.js**

#### **1. Introduction**

* **Project Title:** InsightStream
* **TeamMembers:**
  + Team ID: SWT1D1741164806148446
  + Team Leader: KEERTHANA R - Stu\_keerthana\_r1@cttewc.edu.in
  + Team member: SHAKTHI P - stu\_shakthi\_p@cttewc.edu.in
  + Team member: AISHWARYA M - Stu\_aishwarya\_m1@cttewc.edu.in
  + Team member: ROWFIYA J - stu\_rowfiya\_j@cttewc.edu.in

#### **2. Project Overview**

* **Purpose:** InsightStream aims to transform the way users engage with news by delivering a modern, cozy, and intuitive web application. Built with React.js, the platform provides real-time news updates, categorized content browsing, and personalized search capabilities. The primary goal is to create an accessible and visually appealing experience that empowers casual readers and news enthusiasts to stay informed effortlessly, whether at home or on the go. By integrating real-time data from the NewsAPI and a warm, welcoming design, InsightStream seeks to make news consumption both enjoyable and efficient.
* **Features:**
  + **Real-Time Trending News:** Displays the latest headlines on the homepage with a visually engaging grid layout.
  + **Categorized News Browsing:** Allows users to explore news by categories (e.g., Business, Technology, Health) with dynamic routing and persistent data display.
  + **Advanced Search:** Enables users to search for specific topics or keywords, delivering tailored results in a responsive grid.
  + **Cozy User Interface:** Features a warm color palette, rounded edges, and subtle shadows for a comfortable reading experience.
  + **Newsletter Subscription:** Includes a placeholder form in the footer for future email subscription integration.
  + **Responsive Design:** Ensures seamless functionality and aesthetics across desktop, tablet, and mobile devices.

#### **3. Architecture**

* **Component Structure:** The application follows a modular architecture with a clear separation of concerns:
  + **Root Component (App.js):** Serves as the entry point, integrating Navbar and Footer while managing routing for page components (Home, Category, SearchResults).
  + **Page Components (Pages/):** Home.js renders trending news via Hero, Category.js handles category-specific news with dynamic routing, and SearchResults.js displays search query results.
  + **UI Components (Components/):** Navbar provides navigation and search, Hero showcases trending news, NewsCard renders individual articles, Footer contains layout structure, and Newsletter offers subscription UI.
  + **Interactions:** Data flows from NewsContext to page components, which pass props (e.g., trendingNews to Hero) or render NewsCard instances with article data.
* **State Management:** Utilizes the React Context API through NewsContext.js to centralize state management. This approach simplifies data sharing across components without prop drilling, managing news data, categories, and loading states efficiently.
* **Routing:** Implemented using react-router-dom within App.js. Routes include:
  + / for the homepage (Home).
  + /category/:cat for category-specific pages (Category), where :cat is a dynamic parameter (e.g., "technology").
  + /search for search results (SearchResults). Navigation is seamless, with category links in Navbar triggering route changes.

#### **4. Setup Instructions**

* **Prerequisites:**
  + Node.js and npm: Required for running the React application (download from <https://nodejs.org/en/download/>).
  + Git: For cloning the repository (<https://git-scm.com/downloads>).
  + NewsAPI Key: Obtain a free API key from <https://newsapi.org/> for news data access.
  + Visual Studio Code: Recommended editor (<https://code.visualstudio.com/>).
  + Basic knowledge of HTML, CSS, and JavaScript.
* **Installation:**
  + Clone the repository: Open a terminal and run git clone <https://github.com/Keerthanaminuz/Insightstream>
  + Navigate to the project directory: cd insightstream.
  + Install dependencies: Run npm install to install all required packages listed in package.json (e.g., react, axios, react-bootstrap).
  + Configure the API key: Open src/Context/NewsContext.js in your editor and replace 'YOUR\_API\_KEY' with your NewsAPI key.
  + Verify setup: Start the app with npm start and visit <http://localhost:3000> in your browser to see the homepage.

#### **5. Folder Structure**

* **Client:** The React application resides in the root directory (insightstream/) with the following organization:
  + public/: Static assets and HTML template.
    - index.html: Main HTML file with meta tags for responsiveness and favicon link.
    - favicon.ico, logo192.png, logo512.png: Default React icons (customizable).
    - manifest.json: Configures Progressive Web App (PWA) settings.
    - robots.txt: Allows search engine crawling.
  + src/: Source code directory.
    - Components/: Reusable UI components (Navbar.js, Hero.js, NewsCard.js, Footer.js, Newsletter.js).
    - Context/: State management with NewsContext.js.
    - Pages/: Page-specific components (Home.js, Category.js, SearchResults.js).
    - Styles/: Custom CSS in App.css for theming.
    - App.js: Root component with routing and layout.
    - index.js: Entry point, rendering the app with Context Provider.
    - index.css: Minimal global styles (if needed).
    - reportWebVitals.js: Default CRA file for performance metrics.
* **Utilities:** No standalone utility files are used. API fetching logic is encapsulated in NewsContext.js, and no custom hooks or helper classes are currently implemented.

#### **6. Running the Application**

* **Frontend:** From the project root (insightstream/), run npm start in the terminal. This launches the development server, accessible at <http://localhost:3000>. The app will auto-reload on code changes.

#### **7. Component Documentation**

* **Key Components:**
  + **Navbar.js:** Provides navigation with a search bar and category links. Uses useContext to access categories and useNavigate for routing. No props; relies on global state.
  + **Hero.js:** Displays trending news in a responsive grid. Receives trendingNews (array of article objects) as a prop, passed from Home.js.
  + **Footer.js:** Renders the footer layout with a nested Newsletter component. No props required.
  + **Home.js:** Homepage component rendering Hero. Uses useContext for topNews and loading; no direct props.
  + **Category.js:** Category page with dynamic news display. Uses useParams for the category slug and useContext for fetching; no props.
  + **SearchResults.js:** Displays search results. Uses useContext for searchResults and loading; no props.
* **Reusable Components:**
  + **NewsCard.js:** Renders an individual news article with image, title, description, and link. Prop: article (object with urlToImage, title, description, url). Used across Hero, Category, and SearchResults.
  + **Newsletter.js:** Subscription form with placeholder submit action. No props; self-contained with local handleSubmit function.

#### **8. State Management**

* **Global State:** Managed via NewsContext.js using React Context API:
  + **State Variables:** topNews (trending articles), categories (fixed array: ["business", "technology", etc.]), searchResults (search query articles), loading (boolean for fetch status).
  + **Functions:** fetchTopNews, fetchNewsByCategory, fetchNewsBySearch handle API calls, updating state asynchronously.
  + **Flow:** Components access state via useContext(NewsContext), e.g., Home uses topNews, Category calls fetchNewsByCategory, and Navbar triggers fetchNewsBySearch.
* **Local State:**
  + **Category.js:** categoryNews (array of fetched articles) and isInitialLoad (boolean to prevent flickering) are managed locally with useState.
  + **Navbar.js:** searchQuery (string for search input) is local state, reset after submission.

#### **9. User Interface**

* The UI is designed for coziness and usability:
  + **Home Page:** A grid of trending news articles with images, titles, and descriptions, rendered via Hero.
  + **Category Page:** Displays news for a selected category (e.g., "Technology") in a responsive grid, with persistent data to avoid loading flickers.
  + **Search Results Page:** Shows articles matching the search query, with a fallback message for no results.
  + **Navbar:** Features a search bar and category links, styled warmly and responsively.
  + **Footer:** Includes a newsletter form, centered and inviting.
* (Screenshots or GIFs to be added post-deployment; see Section 12 for placeholders.)

#### **10. Styling**

* **CSS Frameworks/Libraries:**
  + Uses react-bootstrap and bootstrap for responsive layouts (grids, cards, forms) and utilities (e.g., d-flex, me-2).
  + Custom CSS in src/Styles/App.css overrides and enhances Bootstrap with a cozy aesthetic.
* **Theming:**
  + **Color Palette:** Warm gradient background (#f9e7d9 to #f5f5f5), beige navbar/footer (#d9b99b), brown text/buttons (#4a2c1a, #7d4f38), peach cards (#fff8f0).
  + **Design Elements:** Rounded corners (15px on cards, 10px on buttons/inputs), subtle shadows (box-shadow), and hover effects (card lift, button color shift).
  + **Typography:** Uses Georgia font (optional via Google Fonts) for a friendly, readable feel.

#### **11. Testing**

* **Testing Strategy:**
  + Currently, no testing framework is implemented. A recommended strategy includes:
    - **Unit Tests:** Test NewsCard rendering with Jest and React Testing Library.
    - **Integration Tests:** Verify search flow from Navbar to SearchResults.
    - **End-to-End Tests:** Simulate user navigation (e.g., category selection) with a tool like Cypress (not included).
  + To implement: npm install --save-dev @testing-library/react @testing-library/jest-dom jest, then write tests in a \_\_tests\_\_ folder.
* **Code Coverage:**
  + Not currently tracked. Future setup could use Jest’s --coverage flag to ensure 80%+ coverage of key components.

#### **12. Demo**

* **Demo Link:** <https://drive.google.com/drive/folders/1J9NRlxl-71B69EmO7iukBvfkUj6qskkP?usp=drive_link>

#### **13. Known Issues**

* **API Rate Limits:** The NewsAPI free tier has request limits (e.g., 100/day), which may cause fetch failures under heavy use.
* **Newsletter Form:** Limited to a placeholder alert; lacks backend integration for actual subscriptions.
* **Initial Load Delay:** First category load may show a brief delay (1-2 seconds) due to API response time, though mitigated by persistent data display.
* **Image Fallbacks:** Some articles lack images, showing placeholders (<https://via.placeholder.com/150>).

#### **14. Future Enhancements**

* **Article Saving:** Add a "Save Article" button to store articles in local storage or a backend database.
* **Dark Mode:** Implement a theme toggle for a dark, cozy alternative using CSS variables or a library like styled-components.
* **Newsletter Integration:** Connect the form to an email service (e.g., Mailchimp) for real subscriptions.
* **Skeleton Loaders:** Replace "Loading..." with animated placeholders during initial fetches for a smoother UX.
* **Animations:** Introduce fade-in transitions for news cards or page changes using CSS or a library like framer-motion.
* **Enhanced Search Filters:** Add options to filter by date, source, or language via NewsAPI parameters.