**Frontend Development with React.js**

**Project Documentation format**

1. **Introduction**
   * **Project Title**: [Insightstream]
   * **Team Members**:
   * Vijayabharathi K
   * Vasikaran N
   * Vignesh D
   * Vinnarasan S
   * Yogesh M.
2. **Project Overview**
   * **Purpose**:
   * The purpose of InsightStream is to provide users with a fast, responsive, and user-friendly news platform where they can easily browse, search, and read news across categories.
   * **Features**:
   * Real-time news updates, category filtering, search option, responsive design, and bookmarking.

**ArchitectureComponent :**

* + application is built with modular React components:
  + Navbar - Handles navigation across categories.
  + NewsList - Displays a list of news articles.
  + NewsCard - Reusable component for individual article previews.
  + Search Bar - Allows users to search articles.
  + BookmarkList - Shows saved articles.
  + • App - Root component managing routes and state. .
  + **State Management**:
  + Global State: Managed with Context API to handle user preferences, selected category, and bookmarks.
  + Local State: Managed with useState and useEffect inside components for Ul interactions like search input and article loading.
  + **Routing**:
  + The app uses React Router for navigation.
  + / → Home (latest news)
  + /category/:name → Category-based news (e.g., sports, tech, politics)
  + /search → Search results page
  + /bookmarks → User's saved articles.
  + **SetupInstructions**: Prerequisites:
  + Node.js (v16+)
  + npm
  + Git
  + **Installation**:
  + git clone [repository-url]
  + cd insightstream
  + npm install
  + Create a.env file and add:
  + REACT\_APP\_API\_KEY=your\_api\_key
  + REACT\_APP\_BASE\_URL=https://newsapi.org/v2
  + The run:
  + npm start

**Folder Structure** **Client**: Here’s a **short version** for **InsightStream – Folder Structure (Client)**:

### ****Folder Structure – Client:****

insightstream/

┣ src/

┃ ┣ assets/ # Images, icons, and static files

┃ ┣ components/ # Reusable UI components (Navbar, NewsCard, SearchBar)

┃ ┣ pages/ # Page-level components (Home, Category, Search, Bookmarks)

┃ ┣ context/ # Global state management (Context API)

┃ ┣ hooks/ # Custom React hooks

┃ ┣ utils/ # Helper functions and API services

┃ ┣ App.js # Root component with routing

┃ ┗ index.js # Entry point of the app

┣ public/ # Public static assets

┣ package.json

* + **Utilities**:
  + API Service:
  + Handles news API requests and responses.
  + Custom Hooks: For fetching news data and managing bookmarks.
  + Helper Functions: Utilities for date formatting, text truncation, and reusable logic.
  + **Running the Application**:
  + # Start development server
  + npm start
  + # Build for production
  + npm run build
  + **Frontend**: The frontend is built with React.js, using React Router for navigation and Context API for state management. It includes reusable components (Navbar, NewsCard, SearchBar), pages (Home, Category, Search, Bookmarks), and integrates with a news API. Styling is done with [TailwindCSS / CSS Modules].
  + **Component Documentation** **Key Components**: Navbar, NewsList, NewsCard, SearchBar, BookmarkList, App (root component)
  + **Reusable Components**: Button, NewsCard, Navbar, SearchBar - configurable via props for text, styling, and actions.

1. **State Management**:

Global State: Managed with Context API to handle selected category, bookmarks, and user preferences across the app.

Local State: Managed with useState and useEffect for component-level states like search input, loading status, and Ul interactions.

1. **User Interface**
   * 1. Home / Dashboard - Overview of metrics, news, and navigation.
   * 2. News Listing Page - Displays articles with search and filter options.
   * 3. Article Detail Page - Full article view with images and related links.
   * 4. Forms - Login, registration, and submission forms with validation.
   * 5. User Profile / Settings - Edit profile,
   * preferences, and account settings.
   * 6. Notifications / Alerts - Interactive toasts and pop-ups.
   * **Styling**

* **CSS Frameworks/Libraries**:

• Tailwind CSS for utility-first styling.

• Sass for modular custom styles.

* Styled-Components for scoped React component styling. .
* **Theming**:
* Custom Design System - Consistent colors, fonts, and spacing across the app.
* Theming Support - Light and dark mode toggle using CSS variables and Tailwind configuration.
* Reusable Components - Buttons, cards, and form elements follow the design system for consistency.
* **Testing**
* **Testing Strategy**:
* Unit Tests: Jest for individual components.
* Integration Tests: React Testing Library for component interactions.
* End-to-End Tests: Cypress for user workflows and app functionality.
* **Code Coverage**:
* Tool: Jest coverage reports to measure tested lines, functions, and branches.
* Technique: Prioritize critical components, forms, and state logic; identify untested areas using coverage visualization.
* **Screenshots or Demo**
* Provide screenshots or a link to a demo showcasing the application’s features and design.

1. **Known Issues**

* Document any known bugs or issues that users or developers should be aware of.

1. **Future Enhancements**:

•Add new interactive components and charts.

•Improve animations and transitions with Framer Motion.

• Enhance styling, responsiveness, and accessibility.

•Optimize performance with lazy loading and caching.

•Add personalized dashboards and advanced user features.