**Insight Steam News App Documentation**

**Introduction**

**Project Title: Insight Stream**

**Team Members:**

* Sanjay.V -22g202sanjay@gmail.com
* Ganesan .M -22g212ganesan@gmail.com
* Sanjeevi .M -sanjeevi22g228@gmail.com
* Akshaya.G -gakshaya1901@gmail.com

**Project Overview**

**Purpose**

Insight Stream is a dynamic news web application that allows users to fetch and display the latest news from multiple sources. The app features an intuitive interface with a responsive navbar, news category selection, and a real-time search function..

**Features**

1. **Personalized News Feed**
2. **Real-Time News Updates**
3. **Multimedia Support**
4. **In-App Search and Filters**
5. **Local News Coverage**

**Architecture**

**Component Structure**

**1. Client Layer (Mobile/Web App):**

* **User Interface (UI)**
* **Client-Side Logic**

**2. API Layer:**

* **REST or Graph QL APIs**
* **Authentication APIs**:

**3. Backend Layer:**

* **Content Management System (CMS)**:
* **News Aggregator Service**:
* **Recommendation Engine**:
* **Search Engine**:

**4. Database Layer:**

* **Relational Database** (e.g., PostgreSQL, MySQL)
* **NoSQL Database** (e.g., MongoDB, Cassandra):
* **Content Delivery Network (CDN)**:

**5. Real-Time and Notification Service:**

* **Real-Time Updates**:
* **Event-Driven Architecture**:

**State Management**

**State management is crucial for building an efficient and responsive news app like Insight Stream, where multiple components (UI, API, user data, preferences, etc.) need to stay in sync and respond to user actions quickly. The state management solution ensures that changes in the app's data layer (like fetching new news articles, changing user preferences, etc.) are reflected in the user interface.**

**Below is a guide to how state management can be implemented in such a news app:**

**Setup Instructions**

1. **Frontend Setup (React)**

**Step 1: Initialize the Project**

**Step 2: Install Dependencies**

**Step 3: Set up Redux for Global State Management**

**Step 4: Set up Routing with React Router**

1. **Real-Time Features (Socket.IO)**

**3.Deployment**

**Frontend Deployment**

**Real-Time Deployment**

**Folder Structure**

**1.Overall Project Structure**

**2.Frontend Directory (/frontend)**

The frontend is where the user interface and interactions happen. It includes React components, Redux for state management, and services that interact with the backend.

**3.Explanation of Key Files**

**4.Advantages of This Folder Structure**

 Modularity: By separating concerns (backend, frontend, services, controllers, components), you ensure that each part of your application can evolve independently. It makes the code more maintainable and scalable.

 Scalability: The project is structured to handle additional features like authentication, user preferences, or more complex state management.

 Separation of Concerns: Backend logic, API routes, real-time Web Sockets, and frontend logic (UI components, state management) are each isolated, making it easier to manage changes or troubleshoot issues.

 Clean Architecture: This structure follows common best practices for modern web development, ensuring that the app is well-organized and easy to navigate.

**Running the Application**

To start the application locally:

**Running the Frontend (React)**

Live server

npm install

**Component Documentation**

**1. Navbar**

**2. Article Card**

**3. Home Page**

**4. Article Page**

**5. Real Time Updates**

**6. Search Bar**

**7. News Category Selector**

**8. Loader**

**9. Error Boundary**

**State Management**

**User Interface**

1. Why Use Redux for State Management

Redux is a predictable state container for JavaScript apps, particularly useful in React apps with a complex state. It centralizes the state into a global store, which can be accessed by any component. Redux is a good choice for:

2. Key Concepts in Redux

Before we dive into the implementation, it's important to understand these key concepts of Redux:

 **Store**

 **Actions**

 **Reducers**

 **Dispatch**

 **Selectors**

**Styling**

1. Why Use CSS Modules

2. Setup for CSS Modules

1. Create a .module.css file for each component.
2. Import the CSS module in your component file.
3. Use the styles by referring to the class names as properties on the imported CSS module.

3. Example Folder Structure for Styling

Here’s how you can structure your styles:

**CSS Frameworks/Libraries**

* The application uses **Ant Design** for consistent and responsive UI components.

**Theming**

* Custom theming is applied using Ant Design's theming capabilities to align with the application's branding.

**Code Coverage**

* Code coverage is monitored using Jest's built-in coverage tools, aiming for comprehensive test coverage across all modules.

**Screenshots or Demo**

**Live Demo**

<https://drive.google.com/file/d/1ZbnO0rsYpbJaYPOorqntCVCg_z7gRwRO/view?usp=drive_link>=drive\_link

**Screenshots**

<https://drive.google.com/drive/folders/1d1_FGxVph9lhunXuWohpqzJF6m7-W_1L?hl=en_GB>

.

**Future Enhancements**

**1**.Fact-Checking and Misinformation Detection

2.Custom Notifications and Alerts

3. News Subscription Integration

4. Sustainability and Local News

5. Data Privacy and Security

6. Advanced Search Functionality

7. Collaborations with Experts

8. Enhanced User Interface (UI) & User Experience (UX)

9. Subscription or Membership Rewards