Frontend Development with React.js

Project Documentation

Introduction

Project Title: InsightStream - Navigate the News Landscape

Team Id : SWTID1741159570158607

Team Size : 5

Team Leader : MEGALA R-

megamegala057@gmail.com

Team Member : SIVA SANKARI S

poojasri1826@gmail.com

Team Member : SATHYALAKSHMI S

sathyajai1203@gmail.com

Team Member : RITHIKA M

rithikamurugan2412@gmail.com

Team Member : PARAMESWARI G -
param26auguest@gmail.com

Project Overview

Insight Stream is a React-based news application designed to give users access to real-time news from various sources. The application provides a seamless user experience for reading, filtering, and interacting with news articles in different categories like technology, business, politics, and more.

Purposes

- To deliver personalized news based on the user's preferences.
- To ensure a clean, easy-to-use UI/UX for quick navigation.
- To provide in-depth articles with multimedia support such as images, videos, and charts.
- To integrate with news APIs for real-time data fetching.

Features

- Real-time news updates from multiple sources.
- News category selection (e.g., politics, sports, tech).
- Search functionality for specific topics or keywords.
- Article view with options to share or bookmark.
- User profile settings for content personalization.
- Dark mode support for a better reading experience.
- Mobile-responsive design.

Architecture

Component Structure:

The major components of the application include:

- **App**: The root component that contains routing and layout management.
- **Header**: Displays the logo, navigation links, and search bar.
- **NewsList**: Displays a list of news articles filtered by category.
- **NewsItem**: Represents an individual article in the list, showing the headline, description, and image.
- **ArticleView**: Displays the detailed view of a single article with multimedia content.
- **CategorySelector**: Allows the user to choose a news category.
- **SearchBar**: Facilitates the search functionality to find articles by keyword.

State Management:

We use **Redux** for global state management in this project. Redux helps manage the state for the news articles, user preferences, and current article selection.

The state flows through:

- **Redux Store**: Contains the central data, including news data, user preferences, and application settings.
- **Actions & Reducers**: Actions fetch news data, and reducers modify the state.
- **Redux Thunk**: Middleware to handle asynchronous API calls for fetching news data.

For local state management, **React's useState and useEffect hooks** are used within individual components like SearchBar, CategorySelector, and NewsItem.

Routing:

The application uses **react-router-dom** to manage routing between different views:

- /: The homepage with a list of categories.
- /category/:id: Displays news articles from a specific category.
- /article/:id: Displays a detailed view of a single article.
- /profile: User profile page to manage preferences.

Setup Instructions

Prerequisites:

- **Node.js** (v14 or higher)
- **npm** (v6 or higher)

Installation:

1. Clone the repository:
2. git clone https://github.com/username/InsightStream.git
3. Navigate into the project directory:
4. cd InsightStream
5. Install dependencies:
6. npm install
7. Configure environment variables: Create a .env file in the root of the project and set the following variables:
8. REACT_APP_API_KEY=your_news_api_key
9. REACT_APP_API_URL=https://newsapi.org/v2/top-headlines
10. Run the development server:
11. npm start

Folder Structure

Client:

- **public**: Contains the static assets (index.html, icons, etc.)
- **src**:
    - **components**: Reusable UI components like Header, NewsItem, SearchBar.
    - **pages**: Page-level components like HomePage, CategoryPage, ArticlePage.
    - **redux**: Contains Redux store, actions, and reducers.
    - **utils**: Contains helper functions and custom hooks.
    - **styles**: CSS or styled-components for the application styling.

Utilities:

- **helpers.js**: Contains utility functions for formatting dates, managing API requests, etc.
- **useFetchNews.js**: Custom hook for fetching news articles from the API.

Running the Application

To run the frontend locally, execute:

```
npm start
```

The application will be accessible on http://localhost:3000.

Component Documentation

Key Components:

- **App**: The root component responsible for rendering the entire app. Props: None.
- **Header**: Displays the navigation bar and search bar. Props: onSearch function for search functionality.
- **NewsList**: Displays a list of news items. Props: articles (array of article objects).
- **NewsItem**: Represents a single article. Props: article (object containing title, description, and image).
- **ArticleView**: Displays detailed information about a specific article. Props: article (object).
- **SearchBar**: Component for entering search queries. Props: onSearch (function to update search results).

Reusable Components:

- **Button**: Custom button component used across the app. Props: `label`, `onClick`.
- **Card**: A reusable card layout for displaying content. Props: `title`, `content`, `image`.

State Management

Global State:

The global state is managed by **Redux** and includes:

- `news`: Stores the list of fetched news articles.
- `selectedCategory`: Stores the currently selected news category.
- `userPreferences`: Stores user settings such as preferred categories and dark mode status.

Local State:

Local states are managed within components using the `useState` hook. For example:

- `SearchBar`: Stores the search query entered by the user.
- `CategorySelector`: Stores the selected category.

User Interface

The UI is designed to be responsive and user-friendly, with the following key pages:

- **HomePage**: Displays a list of categories.
- **CategoryPage**: Shows articles based on a selected category.
- **ArticlePage**: Detailed view of a single article.

Styling

CSS Frameworks/Libraries:

- **Styled-Components**: Used for creating styled components with dynamic theming.
- **CSS Modules**: Used for scoped styles on individual components.

Theming:

The application supports light and dark modes. The theme is toggled based on the user's preferences and can be controlled using a global state.

Testing

Testing Strategy:

- **Unit Testing**: Components like SearchBar, NewsItem, and Button are tested using **Jest**.
- **Integration Testing**: Ensures that components like NewsList correctly display articles fetched from the API.
- **End-to-End Testing**: Using **Cypress** for testing the user interactions, such as searching for articles or selecting categories.

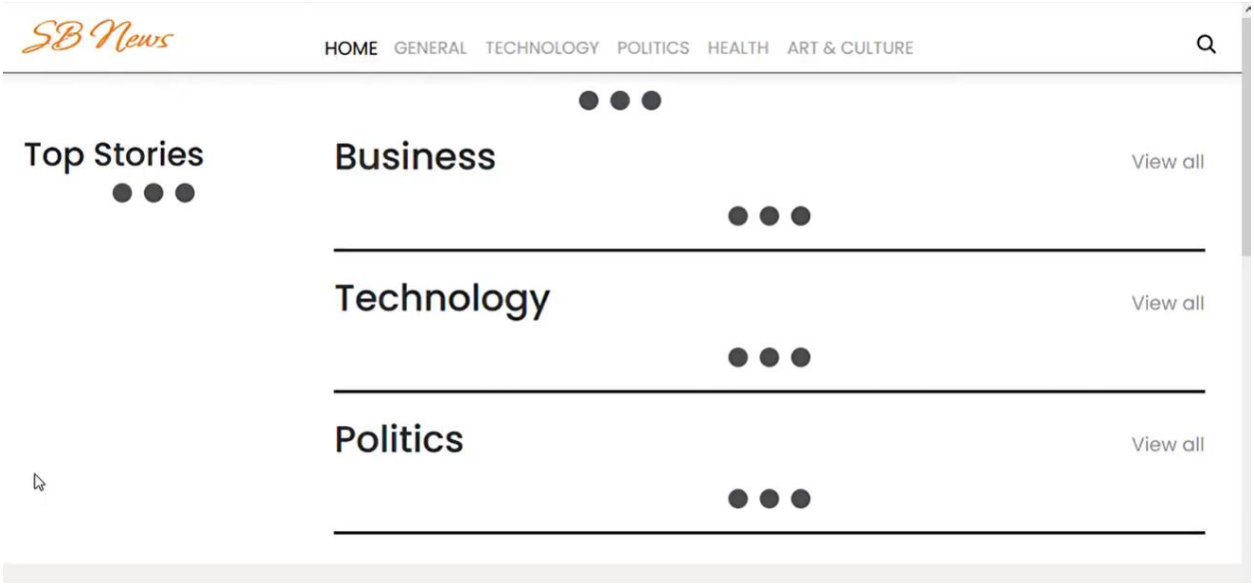
Code Coverage:

We use **Jest** along with **React Testing Library** for coverage. Code coverage is measured with **Istanbul**.

Screenshots or Demo

Screenshots:

1. **Home Page**: Displays news categories.



2. **Article View**: Displays detailed information about a news article.

[Link to demo video or live application : https://drive.google.com/file/d/1imLYjkXnmoAzSYBo80dCNX7I4V-chjkE/view?usp=drivesdkKnown Issues]

- The app may take longer to load news articles due to slow API responses.
- Dark mode may not apply correctly on certain screens.

Future Enhancements

- Add support for push notifications to alert users about breaking news.
- Implement social sharing features to share articles directly on social media.
- Enhance the UI with animations for smooth transitions between pages.