**INSIGHT STREAM: NAVIGATE THE NEWS LANDSCAPE – PROJECT DOCUMENTATION**

1. INTRODUCTION

Project Title:

Insight Stream: Navigate the News Landscape

Team Leader:

Dhananjayan G

Team Members:

1. Deepak S
2. Deepak Raj P
3. Dhanush S

2. PROJECT OVERVIEW

Purpose:

Insight Stream aims to deliver a smooth and engaging frontend experience, helping users easily access and analyse streaming data insights in real time.

Features:

* Real-time data visualization that updates dynamically
* Intuitive and user-friendly interface for effortless navigation
* Responsive design that works well on desktops, tablets, and phones
* Interactive components that let users dive deep into the data

3. ARCHITECTURE

Component Structure:

The app is built with modular React components, organized into:

Pages: Main screens like Dashboard, Reports, and Settings

Components: Reusable UI elements such as charts, tables, and navigation bars

These components communicate through props and callbacks, keeping data flow clear and manageable.

State Management:

We use Reacts built-in hooks like `useState` and `useContext` to handle both local and global state, avoiding the complexity of external libraries like Redux.

Routing:

Navigation between pages is handled smoothly with React Router, allowing users to move around without full page reloads.

4. SETUP INSTRUCTIONS

Prerequisites:

* Make sure you have Node.js (version 14 or newer) installed
* npm (comes with Node.js) for managing packages

Installation Steps:

1. Clone the project repository:

```bash

git clone https://github.com/Dhananjayan007/InsightStream.git

```

2. Go into the project folder:

```bash

cd InsightStream

```

3. Install all necessary dependencies:

```bash

npm install

```

4. If needed, set up environment variables in a `. env` file.

5. FOLDER STRUCTURE

Client (React App):

* `src/components/` — Contains reusable UI components
* `src/pages/` — Holds the main page components linked to routes
* `src/assets/` — Stores images, icons, and other static files
* `src/utils/` — Utility functions and custom hooks to support the app

Utilities:

Helper functions and custom hooks live in `src/utils/`, helping with tasks like data formatting and API calls.

6. RUNNING THE APPLICATION

* To get the app running locally, simply run:

```bash

npm start

```

* This will open the app in your browser at [http://localhost:3000] (http://localhost:3000), and it will reload automatically as you make changes.

7. COMPONENT DOCUMENTATION

Key Components:

* `Dashboard` — Shows live data insights and visualizations
* `ReportList` — Displays a list of reports with filtering options
* `NavigationBar` — Provides easy navigation across the app

Reusable Components:

* `Chart` — A flexible chart component for different data types
* `Button` — A styled button with customizable options
* `Modal` — Popup dialogs for user interactions

8. STATE MANAGEMENT

Global State:

Managed with React Context API, sharing data like user authentication and theme preferences across the app.

Local State:

Individual components handle their own state using `useState` for things like form inputs and toggles.

9. USER INTERFACE

* + The UI is clean and modern, designed for easy navigation.
  + Interactive charts and tables let users explore data in depth.

10. STYLING

CSS Frameworks/Libraries:

The project uses CSS modules and may include Styled-Components for scoped and maintainable styling.

Theming:

Supports light and dark modes, allowing users to switch themes easily.

11. TESTING

Testing Strategy:

We use Jest and React Testing Library to write unit and integration tests, ensuring components work as expected.

Code Coverage:

Coverage reports help track which parts of the code are tested and highlight areas needing more tests.

12. SCREENSHOTS OR DEMO

Coding:A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

13. KNOWN ISSUES

* + Some lint warnings might pop up during development but don’t affect how the app works.
  + The build process may need tweaks if new dependencies are added.

14. FUTURE ENHANCEMENTS

* + Consider adding Redux for more scalable state management.
  + Introduce animations and smooth transitions to improve user experience.
  + Expand testing to 0include end-to-end scenarios.
  + Add more detailed analytics and reporting features.
  + Enhance accessibility and improve mobile responsiveness.