## Frontend Development with React.js

***1*.Introduction:**

Project Title: Insight Steam

Team Members: S.Komathi, K.M.Tejeswani, Y.Ranjini, D.Divya

[Y.Ranjini-program and coding]

[K.M.Tejeswani-voice Video]

[S.Komathi-Documention]

[D.Divya-Phase documentation]

***2.* Project Overview**

**Purpose:**

The **Insight Stream** refers to a continuous flow of valuable data and information that organizations use to gain deeper understanding and drive informed decision-making. It typically involves gathering, analysing, and visualizing insights from various sources such as market trends, customer feedback, or operational performance.

### Type:

### **Data-driven**: Focuses on extracting actionable insights from data.

**Real-time or periodic**: It can be real-time or updated at specific intervals.

### Purpose:

* To provide timely and relevant insights that support strategic decisions.
* To improve operational efficiency, customer experience, or business growth.

### Goals:

* Enable data-driven decision-making.
* Identify emerging trends, opportunities, or risks.

**Features:**

The front-end of a web application is crucial because it’s the part of the application that users interact with directly. Here are some key features and functionalities of the front-end:

1. **User Interface (UI) Design**: Ensuring an appealing and intuitive design that enhances user experience.
2. **Responsive Design**: Making sure the application looks great on all devices, from desktops to mobile phones.
3. **Accessibility**: Ensuring that the application is usable by people of all abilities and disabilities.

***3*.Architecture**

**1. Data Sources**: These are the origins of the data, such as IoT devices, web applications, or social media platforms. The data can be in various formats like JSON or CSV and can be structured or unstructured.

**2. Data Ingestion**: This step involves receiving and storing the data using technologies like Kafka or Estuary Flow streaming pipelines. Basic operations like validation or enrichment can be performed here..

**Major react components**

**1. Functional Components**: These are simple JavaScript functions that return JSX. They are easier to write and understand, and they can use hooks to manage state and side effects.

**2. Class Components**: These are ES6 classes that extend React.Component and include a render method. They are more complex than functional components and are less commonly used in modern React applications.

**State Management:**

### Context API

**Overview**:

* The Context API is a React feature that allows you to pass data through the component tree without manually passing props at every level.
* It is ideal for managing global state or configurations like themes, user authentication, or settings that need to be accessed by various components.

**Use Cases in Insight Stream**:

1. **User Authentication**: Manage and provide user authentication state across the application.
2. **Theme Management**: Handle theme settings (light/dark mode) that need to be consistent across various components.

### Using React Router

**React Router** is the most commonly used library for adding routing capabilities to React applications. It allows you to define multiple routes in your application and navigate between them seamlessly.

**Steps to Use React Router**:

1. **Install React Router**:

bash

npm install react-router-dom

1. **Create Routes**: Define routes in your application using the <Routes> and <Route> components.
2. **Implement Router**: Wrap your application with the <BrowserRouter> component.

### Other Routing Libraries

While React Router is the most popular, there are other routing libraries available:

1. **Reach Router**:
   * Created by the same team that developed React Router.
   * Focuses on accessibility and simplicity.
   * Recently merged with React Router, so most new features will be found in React Router.
2. **Next.js Router**:
   * Built-in routing library for Next.js, a React framework for server-side rendering.
   * Uses a file-based routing system where each file in the pages directory corresponds to a route.

### Choosing a Routing Library

* **React Router**: Best for most React applications due to its flexibility, features, and large community support.
* **Next.js Router**: Ideal for applications built with Next.js, offering built-in support for server-side rendering and static site generation.

**4.Setup Instructions**

### Prerequisites

1. **Node.js**: A JavaScript runtime environment.
   * Download and install from Node.js website.
   * Make sure you also have npm (Node Package Manager) installed, which comes with Node.js..
2. **Git**: A version control system.
   * Download and install from Git website.
3. **Code Editor**: A code editor or Integrated Development Environment (IDE) such as Visual Studio Code.
   * Download and install from Visual Studio Code website.

### Software Dependencies

These are the software dependencies that need to be installed to run InsightStream:

1. **React**: A JavaScript library for building user interfaces.
2. **Redux**: A state management library for JavaScript applications.
3. **React Router**: A routing library for React applications.

#### Installation: 2. Clone the Repository

Open your terminal and clone the InsightStream repository from GitHub:

bash

git clone https://github.com/Ashu1090/InsightStream.git

cd InsightStream

#### 3. Install Dependencies

Navigate to the project directory and install the necessary dependencies using npm:

bash

npm install

#### 4. Configure Environment Variables

Create a .env file in the root directory of the project and add the required environment variables. Here’s an example of what the .env file might look like:

REACT\_APP\_API\_URL=http://your-api-url.com

REACT\_APP\_API\_KEY=your\_api\_key

These variables are used to configure your application, such as API endpoints and keys.

#### 5. Start the Development Server

Start the development server to run the application locally:

bash

npm start

This command will start the server, and you can access the application in your web browser at http://localhost:3000.

#### 6. Build for Production

To build the application for production, use the following command:

bash

npm run build

This will create an optimized build of the application in the build directory.

#### 7. Deploy the Application

Deploy the built application to your preferred hosting service (e.g., Vercel, Netlify, AWS). Follow the hosting service's instructions for deploying a React application.

**5.Folder Structure:**

**Cilent:**

### Folder Structure of the Client Side in InsightStream

The organization of the folder structure in a React application is crucial for maintainability and scalability. Here's a typical folder structure for the client-side of InsightStream, including key folders like components, pages, assets, etc.

InsightStream/

│

├── public/ # Static files

│ ├── index.html # Main HTML file

│ ├── favicon.ico # Favicon

│ └── ... # Other static assets

│

├── src/ # Source files for the application

│ ├── assets/ # Static assets like images, fonts, etc.

│ │ ├── images/ # Images used in the application

│ │ ├── fonts/ # Custom fonts

│ │ └── ... # Other asset types

│ │

│ ├── components/ # Reusable components

│ │ ├── Header.js # Header component

│ │ ├── Footer.js # Footer component

│ │ ├── Sidebar.js # Sidebar component

│ │ └── ... # Other reusable components

│ │

│ ├── pages/ # Page components

│ │ ├── HomePage.js # Home page component

│ │ ├── AboutPage.js # About page component

│ │ ├── DashboardPage.js # Dashboard page component

│ │ └── ... # Other page components

│ │

│ ├── redux/ # Redux-related files

│ │ ├── actions/ # Redux action creators

│ │ ├── reducers/ # Redux reducers

│ │ ├── store.js # Redux store configuration

│ │ └── ... # Other Redux-related files

│ │

│ ├── App.js # Main application component

│ ├── index.js # Entry point of the application

│ ├── routes.js # Application routes

│ └── ... # Other source files

│

└── package.json # Project metadata and dependencies

### Description of Key Folders

1. **public/**:
   * Contains static files that will be served directly by the server.
   * Includes the main index.html file, favicon, and other static assets.
2. **src/**:
   * The main source directory for the React application.
3. **src/assets/**:
   * Contains static assets like images, fonts, and other media files.
   * Organized into subfolders like images/ and fonts/.
4. **src/components/**:
   * Contains reusable React components.
   * Components like Header, Footer, and Sidebar that can be used across multiple pages.
5. **src/pages/**:
   * Contains page components that represent different views or screens of the application.
   * Each file corresponds to a specific page, such as HomePage, AboutPage, and DashboardPage.
6. **src/redux/**:
   * Contains Redux-related files for state management.
   * Subfolders for actions/ (action creators) and reducers/ (reducers).
   * store.js for configuring the Redux store.
7. **src/routes.js**:
   * Defines the routes for the application using React Router or another routing library.
   * Maps URLs to corresponding page components.

### Utilities:

### Utility Classes

Utility classes are similar to helper functions but are organized into classes. These classes contain static methods that can be called without creating an instance of the class. Utility classes are often used to group related functions together. For example, a utility class might contain methods for string manipulation, file handling, or mathematical operations.

### Custom Hooks

In React, custom hooks are functions that allow you to reuse stateful logic across multiple components. Custom hooks start with the word "use" and can be used to encapsulate complex logic that would otherwise be duplicated in multiple components. For example, a custom hook might be used to manage form state, handle API requests, or implement authentication.

**6*.*Running Application:**

To start the frontend server locally for Insight Stream, you can follow these general steps. Note that the exact commands might vary depending on the specific setup of your project, but here's a common approach:

1. **Navigate to the project directory**:

bash

cd path/to/your/project

1. **Install dependencies** (if not already installed):

bash

npm install

1. **Start the frontend server**:

bash

npm start

7.**Componenet Documenatation:**

**Key Components:**

### 1. ****Header****

**Purpose**: The Header component is used to display the top navigation bar of the application.

**Props**:

* title (string): The title to be displayed in the header.
* logo (string): The URL of the logo to be displayed in the header.

### 2. ****Sidebar****

**Purpose**: The Sidebar component is used to display a side navigation menu.

**Props**:

* items (array): An array of navigation items to be displayed in the sidebar.

### 3. ****Footer****

**Purpose**: The Footer component is used to display the footer of the application.

**Props**:

* text (string): The text to be displayed in the footer.
* links (array): An array of footer links to be displayed.

### 4. ****Button****

**Purpose**: The Button component is used to display a clickable button.

**Props**:

* label (string): The text to be displayed on the button.
* onClick (function): The function to be called when the button is clicked.

**Reusable Compoenents:**

### 1. ****Button****

**Purpose**: The Button component is used to display a clickable button.

**Props**:

* label (string): The text to be displayed on the button.
* onClick (function): The function to be called when the button is clicked.
* type (string): The type of the button (e.g., "submit", "button").

### 2. ****Input****

**Purpose**: The Input component is used to display an input field.

**Props**:

* value (string): The value of the input field.
* onChange (function): The function to be called when the input value changes.
* placeholder (string): The placeholder text for the input field.

### 3. ****Modal****

**Purpose**: The Modal component is used to display a modal dialog.

**Props**:

* isOpen (boolean): Whether the modal is open or not.
* onClose (function): The function to be called when the modal is closed.
* children (node): The content to be displayed inside the modal.

### 4. ****Card****

**Purpose**: The Card component is used to display a card with content.

**Props**:

* title (string): The title of the card.
* content (string): The content of the card.
* image (string): The URL of the image to be displayed on the card.

### 5. ****Dropdown****

**Purpose**: The Dropdown component is used to display a dropdown menu.

**Props**:

* options (array): An array of options to be displayed in the dropdown.
* selected (string): The currently selected option.
* onChange (function): The function to be called when the selected option changes.

**8.State Management:**

### . ****Global State:****

Global state refers to the state that is accessible throughout the entire application, no matter where you are in the app. Rather than having different states in each component (which can lead to inconsistencies), global state centralizes this data so that it can be accessed and manipulated from anywhere in the app.

### 3. ****How State Flows Across:****

State flows across an application in a predictable manner using patterns like **unidirectional data flow** or **event-driven architectures**. Here's how it might work in an Insight Stream application:

#### a) ****Initial State Setup:****

* The application starts with an initial global state. This might include user authentication details, data streams, filters, and any other configuration.

#### b) ****Subscriptions and Updates:****

* Components or modules within the application **subscribe** to pieces of the global state they are interested in. For example, a dashboard component might subscribe to the stream of incoming data events or real-time metrics.
* When the state is updated (e.g., a new data event comes in), the **state management system** notifies the subscribed components, which then re-render or take some other action based on the new state.

### Local State:

### 1. ****What is Local State?****

Local state refers to the data and state that is specific to a single component. It exists only within the component's lifecycle and is not shared with other parts of the application unless explicitly passed as props or communicated through events. The main characteristics of local state are:

* **Scoped to a Component**: It is only relevant to the component where it’s defined and used.
* **Short-lived**: Often used to manage temporary values that don’t need to persist beyond the component’s lifecycle (e.g., a toggle switch or a form field value).

### 2. ****Handling Local State in Insight Stream****

In an Insight Stream application, local state is used for scenarios like managing temporary data, UI interactions, or component-specific settings. Here’s how it is typically handled:

#### a) ****State Inside Components:****

* Each component has its own state, stored in a local object or variable that controls its behavior. For example, a **chart component** might maintain the local state of filters applied to the data or the current zoom level.

9.Uscer Interface:

### Principles of UI Design

1. **Simplicity**: Keep the design clean and simple. Minimize clutter and ensure that each element serves a purpose.
2. **Consistency**: Use consistent colors, fonts, and design elements across the interface. This helps users learn and predict interactions.
3. **Feedback**: Provide immediate feedback for user actions. For example, button clicks, form submissions, or loading states should have visual indicators.
4. **Accessibility**: Design with accessibility in mind to ensure all users, including those with disabilities, can interact with the interface effectively.

**10. Styling:**

CSS framework libraries are pre-designed collections of CSS, JavaScript, and other assets that help developers create responsive, consistent, and visually appealing websites and applications quickly. They offer pre-built components like buttons, grids, forms, navigation, and more, reducing the need for developers to build these from scratch.

Here are some popular CSS framework libraries:

### 1. ****Bootstrap****

* **Overview**: One of the most popular CSS frameworks, Bootstrap offers a responsive grid system, pre-designed components, and utilities. It also comes with JavaScript components like modals, dropdowns, and carousels.
* **Website**: [Bootstrap](https://getbootstrap.com/)

### 2. ****Tailwind CSS****

* **Overview**: Tailwind CSS is a utility-first CSS framework that allows developers to build custom designs directly in HTML. It provides low-level utility classes that can be combined to create any design without predefined components.
* **Key Features**:
  + Utility-first approach, offering control over every aspect of styling.
  + Highly customizable using configuration files.
  + Built-in support for responsive design and dark mode.
  + Great for custom designs without needing to override styles.
* **Website**: Tailwind CSS

**11.Testing**

**TestNG** is a popular testing framework for Java-based applications that is widely used for unit testing, integration testing, and end-to-end testing. TestNG supports various testing strategies, allowing flexibility and scalability when testing components. It helps in organizing and running tests efficiently, making it an ideal choice for component-based testing in both unit and integration testing.

Here’s an overview of how TestNG can be used for component testing, including **unit tests** and **integration tests**:

### 1. ****TestNG Basics Overview:****

TestNG provides annotations, a flexible test execution model, and rich reporting, which makes it suitable for different levels of testing:

* **Annotations** like @Test, @BeforeClass, @AfterMethod, etc., control the flow and configuration of tests.
* **Test execution** can be controlled through XML files (testng.xml), enabling batch test execution and test suite management.
* **Parallel test execution** helps speed up the testing process for larger applications.

**12.Known Issues**

<https://github.com/unm1647212205283/InsightStream>

**13. known issues :**

 **Windows 11, version 24H2**: There are several known issues with this update, including problems with AutoCAD 2022 not opening, some ASUS devices failing to install the update, and issues with Dirac Audio causing loss of audio output2.

 **Salesforce**: Known Issues is Salesforce’s bug transparency site, where they share known bugs and workarounds.

14*.* **Future Enhancenents*:***

 **Artificial Intelligence (AI)**: AI is set to revolutionize various industries, from healthcare to finance, by enhancing diagnostics, treatment plans, and data analysis.

 **Machine Learning**: This subset of AI allows systems to learn and improve from experience, impacting fields like marketing and autonomous vehicles.

 **Blockchain Technology**: Beyond cryptocurrencies, blockchain is transforming finance, supply chains, and healthcare by ensuring secure and transparent transactions.