**Insight stream: navigate**

**the news landscape **

|  |  |
| --- | --- |
| TEAM MEMBERS | EMAIL ID |
| Dharshini M | dharshini27212006@gmail.com |
| Deepika N | n.deepikanagaraj2006@gmail.com |
| Bhagyalakshmi S | bhagya151807@gmail.com |
| Abinayasri S | abinayasri072007@gmail.com |

**introduction**

An insight stream is a continuous, real-time flow of actionable insights derived from data as it is generated. This approach, often called "streaming analytics," allows organizations to respond to events as they happen, moving beyond the time-delayed analysis of traditional batch processing.

Introduction to insight streams

The concept of an insight stream shifts the focus of data analysis from looking at historical data to monitoring and reacting to a constant flow of new data. This is particularly critical for applications where timeliness is essential, as the value of the data diminishes over time. Industries such as financial services, telecommunications, and healthcare rely on this process for functions like fraud detection, system monitoring, and patient care.

An insight stream is built on three key ideas:

**PROJECT OVERVIEW:**

* *News Insights is an innovative web application designed for users to explore, organize, and create news. It caters to both beginner and professional insights, providing an intuitive user experience and a vast collection of news insights.*

* Insight Stream automatically creates a live news feed of personalized search insights and important KPIs so you can focus on what's important without digging through data. Insight Stream proactively serves the most relevant information so you can take immediate action on recommendations and drive meaningful results.

**PURPOSE:**

News insights aims to revolutionize the way users interact by offering a seamless platform for discovering, saving, and sharing culinary inspirations. The main goals are: .

**Comprehensive Recipe Management** – Advanced search and categorization for efficient organization.

**Modern Tech Stack** – Utilizing React.js and Rapid API for enhanced functionality.

An Insight Stream's purpose is to automatically deliver personalized, real-time data insights and key performance indicators (KPIs) directly to users, streamlining communication and helping organizations focus on critical information for faster, more effective decision-making. It functions by processing continuous streams of data to identify trends, anomalies, and actionable opportunities as they happen, enabling businesses to respond quickly to emerging situations and improve outcomes.

Key Purposes

* **Real-Time Actionable Insights:**

The primary goal is to provide up-to-the-minute insights from continuous data streams, moving beyond traditional batch processing to enable immediate action.

* **Automated Information Delivery:**

It eliminates the need for users to manually dig through data, instead delivering a live, personalized feed of relevant information.

* **Enhanced Decision-Making:**

By providing current data and highlighting important trends, Insight Streams empower organizations to make timely, well-informed decisions.

* **Streamlined Communication:**

The automated sharing of critical information helps align teams and stakeholders, reducing communication lags and improving collaboration.

* **Focus and Efficiency:**

It helps users focus on what truly matters by filtering out noise and presenting only the most important updates and KPIs, leading to greater efficiency.

* **Proactive Issue Resolution:**

By spotting issues or anomalies as they occur, businesses can respond to critical system failures, security breaches, or other problems before they escalate, minimizing losses.

How It Works (General Concept)

* **Continuous Data Flow:**

Data is processed as it's generated from various sources, such as IoT devices, sensors, and user clickstreams.

* **Real-Time Analysis:**

Streaming analytics engines analyze this continuous flow to identify patterns, deviations, and significant events in real time.

* **Personalized Feed:**

Insights and alerts are then presented in a user-friendly, personalized feed, tailored to the user's specific needs and objectives.

**FEATURES:**

·**News API Integration** – Fetches meals from the MealsDB API.

·**Visual Recipe Browsing** – Image-based navigation of categories.

·**Search Functionality** – Easily find recipes using keywords.

·**Interactive UI** – Built using modern design principles for a smooth experience.

Insight Stream automatically creates a live news feed of personalized search insights and important KPIs so you can focus on what's important without digging through data. Insight Stream proactively serves the most relevant information so you can take immediate action on recommendations and drive meaningful results.

**what are the key features of streams?**

What are the characteristics of a stream? Streams flow downhill due to gravity and the topography defines the direction of flow, as they follow the path of least resistance, through cracks. Small streams, springs and run-off come together to form

larger streams.

Insights influence the decision-making process. They are the foundation on which strategies are built, and they help to

optimise and improve business operations.

**What Are 8 Benefits of Insight Discovery?**

* Improving Communication Skills. Insight Discovery provides a valuable platform for individuals to enhance their communication abilities. ...
* Enhanced Leadership and Decision-Making. ...
* Conflict Resolution and Relationship Building. ...
* Applying Insight Discovery in Your Organisation.

**ARCHITECTURE:**

**Component Structure**

Streaming Data Architecture in 2023: Components & Examples ...An architecture insight stream describes the practical understanding of design principles, spatial qualities, and functional and aesthetic considerations that inform the creation of buildings and environments. This stream encompasses visual imagination, an understanding of the environment and topography, awareness of building history, and knowledge of various architectural styles like classical, contemporary, and industrial. Architects develop this insight by studying design, observing existing structures, and hands-on experience, enabling them to infuse buildings with personality and foster a connection with people.

Key Aspects of an Architecture Insight Stream:

Visual Imagination:

The ability to mentally visualize and manipulate spaces, which helps in designing innovative and suitable solutions from scratch.

Environmental Awareness:

Understanding how buildings interact with their surroundings, considering topography, water features, roads, and other existing structures.

Aesthetic Sensibility:

The capacity to create visually appealing and harmonious buildings that serve expressive purposes.

Functional Design:

Ensuring that buildings fulfill practical requirements while also being aesthetically pleasing.

Understanding of Styles:

Knowledge of various architectural styles, such as classical, contemporary, neoclassical, and industrial.

Contextual Integration:

The art of creating structures that possess a distinctive personality and blend well with their environment.

**How Architects Develop This Insight:**

**Formal Education:**

Studying the principles of architecture, which is the art and technique of designing and building.

**Software and Technology**:

Utilizing design software to realize designs, but always beginning with a conceptual visualization of the space.

Observation and Experience:

Exploring buildings, observing how spaces create different atmospheres, and learning from the real world.

**Creative Problem-Solving:**

Adding innovative ideas and elements to designs, which is an integral part of the architectural process.

The application is divided into three main sections:

* **Pages** – Full-page components (Home, News Category, News Details).
* **Components** – Reusable UI elements (Navbar, Search Bar, News Cards, Category Filter).
* **Styles** – CSS and styling files (Global styles, Component-specific styles).

**Component:**

A common state management example is a user's login status, where a central store holds the isAuthenticated boolean, and components access this state to display different content or buttons for logged-in vs. logged-out users, as demonstrated with React's Context API in Techify Solutions's guide. Another example is a shopping cart, where state management allows the application to add items, update quantities, and reflect these changes in the user interface across multiple components.

1. E-commerce Shopping Cart (Conceptual Example)

Goal: A user can add items to a shopping cart while browsing.

State: The shopping cart's contents (a list of products and their quantities).

How it Works:

When a user clicks "Add to Cart," the application's state management system updates the cart's data.

This change triggers a re-render of the UI, showing the updated cart contents, perhaps in a mini-cart display or a dedicated cart page.

Libraries like Redux or Vuex are used to manage this central cart state, ensuring all parts of the application see the same, updated cart.

2. User Authentication (React Context API Example)

Goal: Show different UI elements for a user based on their login status.

State: An isAuthenticated boolean variable.

How it Works (React Context API):

Create a Context: A AuthContext is created to hold the authentication state.

**AuthProvider**: A provider component (<AuthProvider>) wraps the application, managing the isAuthenticated state and providing it to any child components that need it.

**Consume the Context:** Components like Home use the useContext hook to access isAuthenticated.

**Conditional Rendering:** If isAuthenticated is true, the user sees a "Logout" button and a welcome message; otherwise, they see a "Login" button.

Update State: Clicking "Login" sets isAuthenticated to true, and clicking "Logout" sets it to false, which re-renders the UI accordingly.

**Simple Counter (Flutter StatefulWidget Example)**

**Goal:** A button to increment a counter value displayed on the screen.

State: The count variable, an integer.

How it Works (Flutter)

StatefulWidget: The MyCounter widget is a StatefulWidget because its state (count) can change.

setState(): When the "Increment" button is pressed, the setState() method is called, which increments the count variable.

Re-render: Calling setState() tells Flutter to rebuild the widget, which then displays the new count value.

# State Management :

A common state management example is a user's login status, where a central store holds the isAuthenticated boolean, and components access this state to display different content or buttons for logged-in vs. logged-out users, as demonstrated with React's Context API in Techify Solutions's guide. Another example is a shopping cart, where state management allows the application to add items, update quantities, and reflect these changes in the user interface across multiple components.

1. E-commerce Shopping Cart (Conceptual Example)

Goal: A user can add items to a shopping cart while browsing.

State: The shopping cart's contents (a list of products and their quantities).

How it Works:

When a user clicks "Add to Cart," the application's state management system updates the cart's data.

This change triggers a re-render of the UI, showing the updated cart contents, perhaps in a mini-cart display or a dedicated cart page.

Libraries like Redux or Vuex are used to manage this central cart state, ensuring all parts of the application see the same, updated cart.

2. User Authentication (React Context API Example)

Goal: Show different UI elements for a user based on their login status.

State: An isAuthenticated boolean variable.

How it Works (React Context API):

Create a Context: A AuthContext is created to hold the authentication state.

AuthProvider: A provider component (<AuthProvider>) wraps the application, managing the isAuthenticated state and providing it to any child components that need it.

Consume the Context: Components like Home use the useContext hook to access isAuthenticated.

Conditional Rendering: If isAuthenticated is true, the user sees a "Logout" button and a welcome message; otherwise, they see a "Login" button.

Update State: Clicking "Login" sets isAuthenticated to true, and clicking "Logout" sets it to false, which re-renders the UI accordingly.

3. Simple Counter (Flutter StatefulWidget Example)

Goal: A button to increment a counter value displayed on the screen.

State: The count variable, an integer.

How it Works (Flutter):

StatefulWidget: The MyCounter widget is a StatefulWidget because its state (count) can change.

setState(): When the "Increment" button is pressed, the setState() method is called, which increments the count variable.

Re-render: Calling setState() tells Flutter to rebuild the widget, which then displays the new count value.

**lobal State:**

Managed using React Context API.

**Local State:**

Controlle**G**d via React's useState for component- level updates.

# Routing :

Implemented using React Rout to enable seamless navigation between pages.

Insight stream" is not a standard feature or term in Microsoft Word for document routing. The term "routing" in a business context usually refers to sending a document to different people for review and approval. For this process, Microsoft Word previously offered a feature called a "Routing Slip" (now deprecated), but modern methods include using Microsoft 365's sharing and co-editing features or integrating with platforms like SharePoint for workflows.

**Understanding "Routing" in Word**

**Historical Context:**

In older versions of Word (like WordPerfect, not Word), you could create a "routing slip" to send documents to reviewers and gather comments, as noted in the search results.

**Modern Alternatives in Microsoft Word:**

Sharing and Co-authoring: Instead of routing, you can share the document with colleagues and allow them to collaborate in real-time, which provides a more efficient way to get input.

Comments and Track Changes: Use Word's "Comments" and "Track Changes" features to gather feedback on a document shared with others.

OneDrive/SharePoint: For more formal workflows, save your document to OneDrive or a SharePoint site, where you can use more advanced document management and workflow features.

What "Insight Stream" Might Mean:

Dynamic Data: It might refer to dynamic data, like information pulled from a system or database, that appears within the document. This is different from document routing.

Business-Specific Terminology: "Insight stream" could be a custom term used within a specific company or software system to describe a particular type of document or process, not a Word-native function.

How to Route Documents (Using Modern Methods)

Share Collaborate: Use the Document: Click the "Share" button in Word to send a link to colleagues.

the built-in features for real-time co-authoring, comments, and Track Changes to get feedback.

Use a Workflow System: If you need a formal routing process, consider integrating Word with Microsoft Power Automate or SharePoint for a more robust workflow solution.

**SETUP INSTRUCTIONS:**

To set up instructions from an "insight stream" (likely a reporting or data-generation platform like OneStream Documentation) into a Word document, you will need to export the instructions or data as a variable string from the insight stream platform and then insert this variable into your Word document using the Field dialog box under the DocVariable name, making sure to paste the string correctly and press Alt+F9 to view or hide field codes.

Step 1:

Get the Variable String

In the "insight stream" platform, find the setting for "Extensible Document Settings" or similar.

Look for "Insert Content Using Document Variables" and expand the option for "Microsoft Word Document".

Select the line of syntax that represents your instructions or data.

Click the Copy to Clipboard button to copy the entire string.

Step 2:

Insert the Variable in Word

Open Microsoft Word, either a new document or an existing one.

Place your cursor where you want the "insight stream" content to appear.

Go to the Insert tab on the ribbon, then select Quick Parts > Field.

In the "Field" dialog box, under "Field Names," select DocVariable.

In the "New name" field, paste the string you copied in Step 1.

Click OK to insert the field into your document.

Step 3:

Process and View the Content

For the content to display correctly, you may need to update the field by pressing Alt+F9 (or Alt+Fn+F9 if Alt+F9 doesn't work) to toggle between showing and hiding field codes.

If the expected content does not display, try again after pressing Alt+F9.

You may also need to manually update the backslash (`\`) symbols in the field code for the Document Variable to process correctly.

# Prerequisites:

# To create a Word document for prerequisites within an insight stream, you must first determine the specific "insight stream" you are referring to, as the term can apply to different platforms like Microsoft StreamInsight, Power BI, Salesforce, or others. After identifying the platform, gather its specific requirements (e.g., data sources, software, configurations) and document them in the Word file using clear sections like System Requirements, Data Sources, Configuration Steps, and User Permissions.

# 1.

# Identify the "Insight Stream" Platform

# The term "insight stream" is broad. You need to clarify which system you're working with.

# Common examples include:

# Microsoft StreamInsight A deprecated platform for real-time data analysis, primarily used in IoT applications.

# Power BI: For setting up real-time streaming data in dashboards.

# Salesforce:

# For creating and managing streaming insights for various applications.

# Other platforms: Like Pega or cloud services that offer data streaming and analysis capabilities.

# 2. Gather Specific Prerequisites

# Once you know the platform, find its specific requirements:

# System Requirements Information on hardware, operating systems, and software versions needed.

# Data Source Requirements: Details on the types of data, connection details, and formats.

# Software & Tools: Any required libraries, SDKs, or development tools.

# User Permissions: Necessary roles and access levels for setting up and operating the stream.

# Network Configuration: Specific port numbers or firewall settings if applicable.

# 3. Document in a Word Document

# Create a structured Word document:

# Title: Clearly label the document, e.g., "Prerequisites for [Platform Name] Insight Stream."

# Introduction: Briefly explain the purpose of the insight stream and the document.

# Sections Organize the gathered information into logical sections:

# System Requirements: Hardware, OS, software versions.

# Data Sources: Types, formats, and access details for incoming data.

# Software & Tools: Required libraries, SDKs, development environments.

# Configuration: Step-by-step instructions for setting up the stream.

# User Permissions: Required roles, accounts, and access levels.

# Visuals: Include diagrams or screenshots to illustrate complex steps or configurations.

# Version Control: Use Word's revision features or include a version history at the start of the document.

# Review and Approval: Get the document reviewed and approved by stakeholders

**Node.js & npm** – Install from [Node.js website.](https://nodejs.org/)

**React.js** – Set up a new project using npx **createreact-app my-react-app**

**cd my-react-app npm**

**Start**

# Installation Steps

To install and configure an "Insight stream" within a Word document, you must first clarify which "Insight" product you are referring to, as "Insight" is a common term used by various software vendors for different products like UiPath Insights, DUG Insight, or Microsoft's Viva Insights. Once the specific product is identified, you must find the corresponding official documentation, which will contain the correct installation and configuration steps for that product, as the process is entirely dependent on the software.

General Steps to Find the Correct Instructions

1. Identify the Specific "Insight" Product:

Determine the full name and vendor of the "Insight" software you are using, for example, "UiPath Insights" or "DUG Insight".

2. Locate the Official Documentation:

Visit the vendor's official documentation website or support portal for that specific "Insight" product.

3. Find the Installation Guide:

Search the documentation for sections titled "Installation," "Getting Started," or "Setup".

4. Review the Configuration Steps:

After installation, follow the product-specific instructions for configuration, which often involve setting up connections to other services, databases, or users.

Example for UiPath Insights

Download the Installer:

Download the relevant InsightsInstaller.msi file from the UiPath Documentation Portal.

Run the Installer:

Use the command prompt to run the installer with administrative privileges: msiexec /i InsightsInstaller.msi.

Follow the Setup Wizard:

The wizard will guide you through prerequisite checks and the configuration of server URLs, Orchestrator settings, database connections, and email settings.

Example for DUG Insight

Download the Installer: Download the installer for the DUG Insight product.

Install the Software: Run the installer and follow the on-screen instructions in the Setup Wizard to complete the installation.

Launch and Configure: After installation, launch the program from the shortcut or start menu to access its configuration options.

1. **Clone the repository** git clone [repository URL]
2. **Navigate into the project directory** cd recipe-app-react
3. **Install dependencies** npm install
4. **Set up environment variables** (if required) by creating a .env file and adding necessary API keys.
5. **Start the development server** npm start
6. **Access the application**

Open [http://localhost:3000](http://localhost:3000/) in your web browser.

# FOLDER STRUCTURE :

The core WordPress folder structure consists of three main directories and several important files in the root directory:

Root Directory:

wp-admin:

Contains all the files and folders that power the WordPress administration interface. These files are responsible for the backend functionality of your site.

wp-content:

This is where all user-generated content and extensions reside. This includes:

**plugins:**

Stores all installed plugins.

**themes:**

Stores all installed themes.

**uploads:**

Stores all media files uploaded to the WordPress Media Library, typically organized by year and month.

**languages:**

Stores language translation files.

**Wp includes**:

Contains the bulk of the core Word Press files, including essential PHP files, JavaScript, and CSS files required for Word Press to function. This directory also houses common functionalities like the database API, HTTP API, and plugin API.

**Important Files in the Root Directory:**

index.php:

The main entry point for your Word Press site, initiating the loading process.

Wp -config. php:

A crucial configuration file containing database connection details, security keys, and other important settings.

.htaccess:

A server configuration file (on Apache servers) that manages permalinks, redirects, and other server-level settings.

wp-login.php:

Handles user login and authentication.

**Key Considerations:**

Core Files:

The wp-admin and wp-includes directories contain the core WordPress files and should generally not be directly edited, as modifications can be overwritten during updates.\

Customization:

Most WordPress development and customization, such as creating themes or plugins, occurs within the wp-content directory.

Content Storage:

Posts, pages, and other content are stored in the WordPress database, not as individual files within the folder structure

The project is structured into different directories for better organization and maintainability.

Below is an overview of the folder structure.

The core WordPress folder structure consists of three main directories and several important files in the root directory:

Root Directory:

wp-admin:

Contains all the files and folders that power the WordPress administration interface. These files are responsible for the backend functionality of your site.

wp-content:

This is where all user-generated content and extensions reside. This includes:

plugins: Stores all installed plugins.

themes: Stores all installed themes.

uploads: Stores all media files uploaded to the WordPress Media Library, typically organized by year and month.

languages: Stores language translation files.

wp-includes:

Contains the bulk of the core WordPress files, including essential PHP files, JavaScript, and CSS files required for WordPress to function. This directory also houses common functionalities like the database API, HTTP API, and plugin API.

Important Files in the Root Directory:

index.php:

The main entry point for your WordPress site, initiating the loading process.

wp-config.php:

A crucial configuration file containing database connection details, security keys, and other important settings.

.htaccess:

A server configuration file (on Apache servers) that manages permalinks, redirects, and other server-level settings.

wp-login.php:

Handles user login and authentication.

Key Considerations:

Core Files:

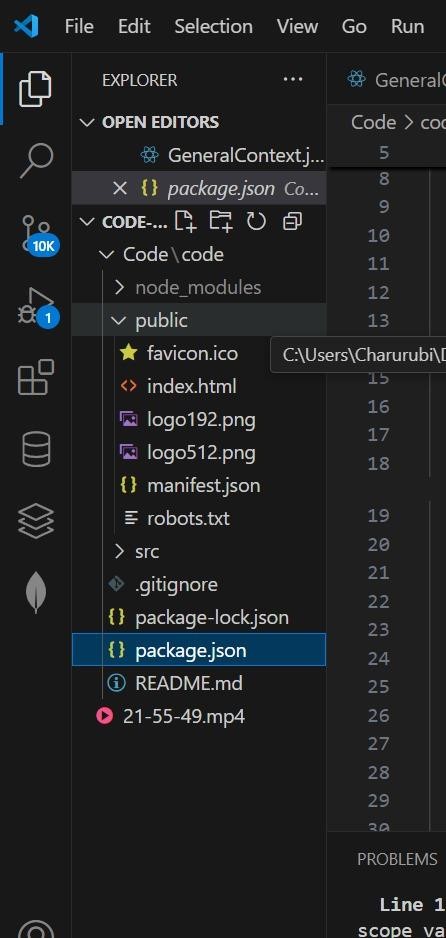
The wp-admin and wp-includes directories contain the core WordPress files and should generally not be directly edited, as modifications can be overwritten during updates.

Customization:

Most WordPress development and customization, such as creating themes or plugins, occurs within the wp-content directory.

Content Storage:

Posts, pages, and other content are stored in the WordPress database, not as individual files within the folder structure



**Client**

The "client structure" in WordPress can refer to two main concepts:

The structure of a WordPress website built for a client:

This encompasses the organization and planning involved in delivering a complete website solution. It includes:

Project Planning: Defining scope, requirements, timeline, and communication plans with the client.

Website Architecture: Designing the site's navigation, content hierarchy (pages, posts, categories, tags), and overall user experience.

Technical Implementation: Choosing themes, plugins, and potentially custom code to achieve desired functionalities.

Content Management: Planning for content creation, migration, and ongoing updates.

Client Handoff: Providing training, documentation, and necessary access credentials.

Post-Launch Support: Offering maintenance, updates, and troubleshooting.

The structure of a "client portal" within a WordPress website:

This refers to a specific, often private, section of a WordPress site designed for clients to access resources, communicate, or manage their projects. Key elements of a client portal structure include:

User Management: Creating dedicated user roles and permissions for clients, controlling their access to specific content or functionalities.

Content Restriction: Utilizing plugins or custom code to restrict certain pages, posts, or media files to authenticated clients only.

Resource Sharing: Providing a centralized location for clients to access documents, files, invoices, or project updates.

Communication Tools: Integrating features like private messaging, feedback forms, or support ticket systems.

Customization: Tailoring the portal's appearance and functionality to align with client needs and branding.

Both interpretations emphasize organization and planning, whether for the entire website delivery process or for a dedicated client-facing area within the site.

**Dive deeper in AI Mode:**

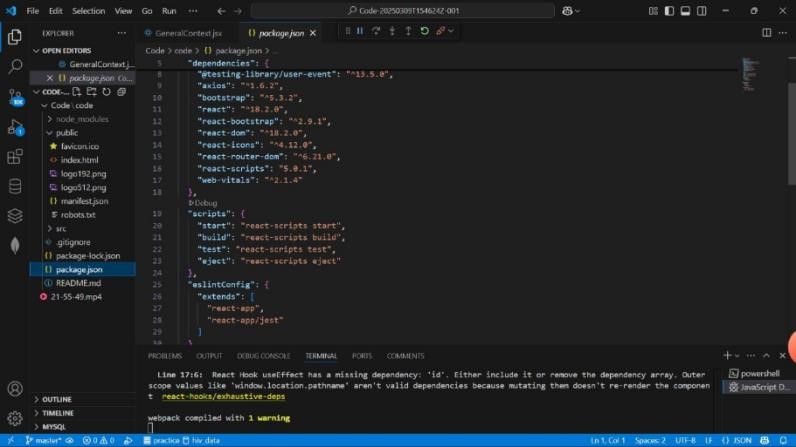
The **client** folder (inside src/) contains the core files of the frontend application. It consists of:

·**Components:** Houses all reusable UI components such as buttons, cards, and the navigation bar.

·**Pages:** Contains full-page components, including the homepage, recipe details page, and category listings.

.**Styles:** Includes all CSS or SCSS files to style the application.

.**Assets:** Stores images, icons, and other static files used throughout the app. Example structure inside src/:



**UTILITIES:**

WordPress utility plugins add essential, often specific, functionalities to a website, ranging from performance and security (like WP Rocket for caching and LiteSpeed Cache) to content creation and management (like Elementor for page building or TablePress for tables) and administrative tasks (such as Loco Translate for translating content or All-in-One WP Migration for site backups). These plugins are available on the official WordPress.org plugin repository and third-party marketplaces like CodeCanyon and Envato Market.

Common Types of WordPress Utilities

Performance & Security:

Enhance site speed and protect it from threats.

WP Rocket: Improves website speed by caching and other performance optimizations.

LiteSpeed Cache: An all-in-one caching and optimization solution for LiteSpeed web servers.

WP Ultimate Firewall: A firewall plugin for security enhancements.

Content & Design:

Help create and manage content more effectively.

Elementor: A page builder to visually design and manage your website.

TablePress: A plugin for creating and managing tables on your WordPress site.

Yellow Pencil: A live CSS editor that allows for easy visual customization.

Administration & Management:

Streamline backend operations and site maintenance.

Loco Translate: Translates themes and plugins directly within the WordPress admin area.

All-in-One WP Migration: A tool for backing up and migrating WordPress sites.

MyBrain Utilities: A general utility plugin offering various tools to manage a WordPress site.

Slugs Manager: Helps clean up old and unused permalinks from the WordPress database.

Where to Find WordPress Utilities

WordPress.org Plugin Repository:

The official source for free WordPress plugins, including many utility tools.

Third-Party Marketplaces:

Websites like CodeCanyon and Envato Market offer a wide range of premium (paid) utility plugins.

How to Install a WordPress Utility Plugin

Log in: to your WordPress Dashboard.

Navigate to Plugins > Add New.

Search: for the utility plugin you need.

Click Install Now and then Activate Plugin.

**RUNNING THE APPLICATION:**

An "insight stream" can refer to a continuous feed of data or a software feature providing real-time analysis and actionable information, with common examples including Application Insights for software monitoring, [Conductor's Insight Stream](https://www.google.com/search?sca_esv=6171a225f406e3e6&cs=0&q=Conductor%27s+Insight+Stream&sa=X&ved=2ahUKEwieuMyVh9WPAxVm3TgGHULwK00QxccNegQIAhAC&mstk=AUtExfCTEqyC353hyldFvoaKwXoW5TVROukpTdBnL2p-u8BePIwvZfvWuSLJQpTAjICikTcWH-C6Z7_-5OB6WMVWHKdRAXOvyx_LztqIBRcfzdGBuVAkRg_9dm4dO8e9V7LhXHk&csui=3) for SEO analysis, and Askable's Insight Streams for continuous user feedback. The specific application depends on the context of the "insight stream" you are referring to.

Here are some common types of "insight streams":

* [**Application Insights**](https://www.google.com/search?sca_esv=6171a225f406e3e6&cs=0&q=Application+Insights&sa=X&ved=2ahUKEwieuMyVh9WPAxVm3TgGHULwK00QxccNegQINRAC&mstk=AUtExfCTEqyC353hyldFvoaKwXoW5TVROukpTdBnL2p-u8BePIwvZfvWuSLJQpTAjICikTcWH-C6Z7_-5OB6WMVWHKdRAXOvyx_LztqIBRcfzdGBuVAkRg_9dm4dO8e9V7LhXHk&csui=3)**(Microsoft Azure):**
  + **Purpose:** An observability service that collects telemetry data from applications to monitor their performance, usage, and availability.
  + **Features:** Provides real-time metrics (like the Live Metrics Stream) and usage analytics to help developers understand user behavior, identify bottlenecks, and troubleshoot issues.
  + **Usage:** Monitor live applications, see what features are popular, and ensure users are completing their goals.
* [**Insight Stream (Conductor):**](https://www.google.com/search?sca_esv=6171a225f406e3e6&cs=0&q=Insight+Stream+%28Conductor%29%3A&sa=X&ved=2ahUKEwieuMyVh9WPAxVm3TgGHULwK00QxccNegQINBAC&mstk=AUtExfCTEqyC353hyldFvoaKwXoW5TVROukpTdBnL2p-u8BePIwvZfvWuSLJQpTAjICikTcWH-C6Z7_-5OB6WMVWHKdRAXOvyx_LztqIBRcfzdGBuVAkRg_9dm4dO8e9V7LhXHk&csui=3)
  + **Purpose:** A feature within Conductor's SEO platform that delivers a live, personalized feed of important search data and KPIs.
  + **Features:** Focuses on providing actionable recommendations, tailoring insights to user goals, and facilitating team communication about those insights.
  + **Usage:** Catch opportunities, take immediate action on data, and improve decision-making in search engine optimization.
* [**Insight Streams (Askable):**](https://www.google.com/search?sca_esv=6171a225f406e3e6&cs=0&q=Insight+Streams+%28Askable%29%3A&sa=X&ved=2ahUKEwieuMyVh9WPAxVm3TgGHULwK00QxccNegQINxAC&mstk=AUtExfCTEqyC353hyldFvoaKwXoW5TVROukpTdBnL2p-u8BePIwvZfvWuSLJQpTAjICikTcWH-C6Z7_-5OB6WMVWHKdRAXOvyx_LztqIBRcfzdGBuVAkRg_9dm4dO8e9V7LhXHk&csui=3)
  + **Purpose:** A platform providing continuous, high-quality, and strategic user insights.
  + **Features:** Generates ongoing streams of user feedback that can be immediately applied across multiple business functions.
  + **Usage:** Gather continuous insights at scale to inform product development and other strategies.

**To start the frontend server, run: npm start**

**Then, open http://localhost:3000 in your browser.**

## COMPONENT DOCUMENTATION KEY COMPONENTS

Insight Stream documentation would cover components related to live, streaming data, such as data connectors for ingestion, real-time processing engines, storage for the data, and analytical tools to derive insights from the stream. Key documentation components would detail how the stream collects and processes data, including the specific streaming data architecture and its components, real-time analytics, and the personalized insights and KPIs it generates.

Key Components and Documentation Areas for Insight Stream

* **Data Ingestion & Sources:**
  + [**Connectors**](https://www.google.com/search?sca_esv=6171a225f406e3e6&cs=0&q=Connectors&sa=X&ved=2ahUKEwjrsNjNh9WPAxV71jgGHR0WMLcQxccNegQIFBAB&mstk=AUtExfC18Sttancx0yV4xmc4QzgR3hl9-NOJ003foJY5fvKwc7FuG3LcsUOSN7WBH4cjfuHyZWfMf-r1T2mS-hZFSYFugBSBHuV1xJ8OADyLk4R76wOxC2z4OlPtfQnnrL9VLZI&csui=3)**:** Documentation should detail how Insight Stream connects to various data sources to collect real-time data.
  + **Data Format:** Information on the types of data that can be ingested and how it's formatted for streaming.
* **Real-Time Data Processing:**
  + **Streaming Engine:** How data is processed immediately as it is generated, focusing on low-latency.
  + **Real-Time Analytics:** Documentation on how the system performs immediate analysis and derives insights from the stream.
* **Data Storage:**
  + **Persisted Storage:** Details on where and how the streaming data is stored for future use or analysis.
* **Insights & KPIs:**
  + **Personalized Feed:** Documentation explaining how the stream automatically creates a live, personalized feed of insights and key performance indicators.
  + **User Focus:** How the system prioritizes and presents information to help users focus and take action.
* **Architecture & Management:**
  + **Streaming Data Architecture:** Explanation of the underlying architecture that supports the real-time data flow.
  + **User Interface:** How users interact with the system to manage settings, view data, and receive insights.

**Purpose:**

Provides site-wide navigation to help users access different sections like Home, Categories, and Search**.**

**Props:**

* logo (string): Path to the logo image displayed in the navbar.
* menuItems (array): List of navigation items such as "Home", "Categories", and "Search".
* onSearchSubmit (function): A callback function triggered when the search form is submitted.

## 2. Hero Component (Hero.js)

**Purpose:**

* Displays an introduction to the site and a call-to-action (CTA) to engage users.
* Contains a button to encourage users to explore more content (e.g., "Explore Latest News").

**Props:**

* title (string): Main heading or introduction for the hero section.
* ctaText (string): Text on the call-to-action button (e.g., "Explore").
* ctaLink (string): URL or path for the CTA button to redirect users

## 3. News List Component (NewsList.js)

**Purpose:**

* Displays a list of news articles, filtered based on categories, search results, or user preferences.

**Props:**

* articles (array): List of news article objects (e.g., title, description, publication date).

## 4. News Card Component (NewsCard.js)

**Purpose:**

* Represents a single news article preview with a title, short description, image, and link to the full article.

**Props:**

* image (string): URL for the image associated with the article.
* title (string): Title of the news article.
* description (string): A brief description of the article.
* articleLink (string): URL to view the full article.

## 5. Categories Component (Categories.js)

**Purpose:**

* Displays a list of categories (e.g., Sports, Politics, Technology) to filter news articles based on users' interests.

**Props:**

* categories (array): List of categories (e.g., ["Politics", "Technology", "Sports"]).
* onCategorySelect (function): Callback function to filter news based on selected category**.**

## REUSABLE COMPONENT

Documentation for reusable components in an "insight stream" context refers to a comprehensive guide for a library of components designed to display and interact with analytical data. For these components to be truly reusable across various dashboards and projects, clear and detailed documentation is essential.

Key elements of insight stream component documentation

A robust component library for data insights requires detailed documentation covering the following areas:

Component catalog

This is a high-level list or index of all available components, often including a visual snapshot of each one.

* Purpose: A concise description of what each component does.
* Examples: Live, interactive examples that demonstrate the component in action, showcasing different variants and use cases.

API documentation

This section provides a formal specification for each component's properties, events, and slots.

* Props: A list of all configuration options (props) the component accepts, including their data type, a brief description, and whether they are required.
* Events: Details on the custom events the component can emit, including the payload data they carry.
* Slots or ng-content: For frameworks that support it (like Vue or Angular), documentation on content projection slots for adding custom content into the component.
* Bundlesize and dependencies: Information on the component's footprint and its required dependencies can be helpful for performance-conscious developers.

Usage guidelines

To ensure consistent use, the documentation should include best practices and practical advice.

* Example code: Copy-and-paste code snippets showing how to implement the component in different scenarios. For example, a DataTable component might show examples with different column configurations.
* Best practices: Recommendations on how and when to use the component effectively and how to avoid anti-patterns.
* Theming and styling: Clear instructions on how to customize the component's appearance, often with options for size, color variants, or custom styling.

Accessibility and testing

A professional component library includes guidance on how to use the components to build accessible user interfaces.

* WCAG compliance: Information on how the component meets Web Content Accessibility Guidelines (WCAG).
* Testing instructions: The documentation should cover how to write individual tests for components and may include references to testing tools.

Versioning and release notes

For a dynamic library, tracking changes is crucial for team members.

* Versioning: Each component should be versioned independently to allow for more granular updates.
* Release notes: A log of changes, new features, and bug fixes for each version of the component.

Benefits of thorough component documentation

Investing in documentation for reusable components provides significant benefits for development teams.

* Improved usability: Makes it easier for other developers to discover, understand, and use the components, reducing the learning curve.
* Increased productivity: Teams can spend less time reinventing the wheel and more time focusing on delivering new insights.
* Consistency: Ensures that UI elements and metrics are formatted consistently across different reports and dashboards, which is vital for data interpretation.
* Reduced errors: Comprehensive documentation reduces misunderstandings and misaligned implementation, preventing a disorganized final product.
* Scalability: Facilitates scaling development across teams and departments by providing a centralized source of truth for component usage.

### 1. Search Bar Component (SearchBar.js)

**Purpose:**

* Allows users to search for news articles by keywords.

**Props:**

* onSearch (function): Callback function to handle the search query submitted by the user.
* placeholder (string): Placeholder text for the search input (e.g., "Search news articles...").

### 2. Button Component (Button.js)

**Purpose:**

* A reusable button that can be used for various actions across the site like submitting forms, triggering events, or navigating.

**Props:**

* text (string): The text that will appear on the button (e.g., "Submit", "Search").
* onClick (function): Callback function that executes when the button is clicked.
* style (string): A string for button style (e.g., "primary", "secondary").
* disabled (boolean): A flag to disable the button when set to true.

### 3.Card Component (Card.js)

**Purpose:**

* A reusable card layout for displaying content such as articles, products, or services.

**Props:**

* title (string): The title to be displayed on the card.
* image (string): The URL of the image to be displayed in the card.
* description (string): A short description of the content.
* link (string): A URL to link the user to the full content.

### 4.Modal Component (Modal.js)

**Purpose:**

* A reusable modal dialog to display content like confirmation, notifications, or forms without navigating away from the current page.

**Props:**

* isOpen (boolean): A boolean value that controls whether the modal is visible.
* title (string): The title to be displayed in the modal header.
* children (node): The content inside the modal (could be text, forms, or other components).
* onClose (function): A callback function to close the modal.

### STATE MANAGEMENT

State management in News Insights (or any web application) refers to how the application handles and stores data that influences the user interface and other parts of the app. This includes managing things like user authentication, displayed articles, search results,State management in **NEWS INSIGHTS** application ensures efficient data handling across different components. The project incorporates both global state for shared data and local state for componentspecific interactions.

### GLOBAL STATE

In a modern web application like News Insights, global state management refers to storing and managing state that can be accessed across multiple components. This allows data to be shared between different parts of the application without the need to pass props manually through each component. For large-scale applications, global state management is essential to avoid prop-drilling and ensure better scalability and maintainability.

### Global State Usage in News Insights

* User Authentication:Stores user login status (isAuthenticated) and user details (user).
* Search Results: Holds the searchQuery and articles fetched based on the query.
* Loading State:Tracks the loading status (loading) while fetching articles.

**How Local State Works in Components:**

**Usage:** Local state is typically used for things like form input values, toggle states, or visibility of UI elements.

**Encapsulation:** Local state is isolated within a component and not shared with other components.

**State Updates:**State is updated using the update function (e.g., setState).React re-renders the component when the state changes to reflect the updates in the UI.

### User Interface

The User Interface (UI) of the News Insights project is designed to provide a clean, intuitive, and user-friendly experience for users interacting with the application. The UI is focused on accessibility, ease of navigation, and delivering the latest news in an organized and attractive layout.

**1. Navigation Bar (Navbar)**

* Purpose: Provides easy access to different sections of the app.
* Components:
* Logo on the left side.
* Navigation links: Home, Categories, Search, Favorites, Profile, Login/Logout.
* Responsive design for mobile and desktop views.
* Dropdown menu for categories or user profile options.

**2. Hero Section**

* Purpose: Serves as the main introduction or welcome area of the app.
* Components:
* Headline text or call-to-action (CTA) inviting users to explore news.
* A search bar for users to quickly find news topics.
* Featured images or news headlines to draw attention to important stories.

**3. Search Bar**

* Purpose: Allows users to search for articles by keywords or topics.
* Components:
* Input field for the search term.
* Search button to trigger the query.

Auto-suggestions or recent searches for quick access.

**4. Article List / News Feed**

* Purpose: Displays a list of news articles based on search query or category.
* Components:
* Card-like layout for each article displaying title, description, image, and a link to read more.
* Pagination controls at the bottom to navigate through multiple pages of results.
* "Load More" button for infinite scrolling or fetching more articles.

**5. Article Detail Page**

* Purpose: Provides in-depth content for a selected article.
* Components:
* Article title, author, and publication date.
* Full article content with images, videos, and text.
* Social sharing buttons (Facebook, Twitter, etc.).
* Option to mark as a favorite or save the article.

### *STYLING*

Styling plays a key role in enhancing the visual appeal and user experience of the News Insights project. The goal is to create a clean, modern, and responsive design that provides easy navigation, smooth interactions, and an intuitive layout. Below is an overview of how the styling is applied in the project.

**1. General Styling Approach**

* **CSS Frameworks:**
* Bootstrap or Tailwind CSS can be used to accelerate the development of responsive layouts and UI components.
* For custom designs, plain CSS or CSS-in-JS (e.g., styled-components) can be employed to fine-tune the application.

* **Responsive Design:**
* The app should be fully responsive, ensuring it works seamlessly across devices like mobile phones, tablets, and desktops.
* Media queries are used to adjust layout, font sizes, and other elements to fit different screen sizes.

* **Color Scheme:**
* The design should feature a modern, professional color palette with good contrast for readability. Examples:
* Primary color: A bold color for the logo and key elements (e.g., blue, green).
* Secondary colors: Lighter shades for backgrounds and borders (e.g., light grey, soft blue).

**Text colors:** Dark text for readability on light backgrounds and white text for dark sections.

#### 2. Styling Key Components

**a. Navbar Component:**

* Purpose: Provides site-wide navigation.
* Styling:
* Fixed position on top for easy access to the navigation links.
* Background color can be dark with white text for high contrast.
* Hover Effects on navigation links for better interactivity.
* Dropdown menus for categories or user profile, styled with hover transitions.

css

/\* Navbar Styling \*/

.navbar {

background-color: #003366; padding: 10px 20px; position: fixed; width: 100%; top: 0; z-index: 100;

}

.navbar a { color: #ffffff; margin: 0 15px; text-decoration: none; font-size: 18px;

}

.navbar a:hover { color: #ffcc00;

}

**b. Hero Section:**

* Purpose: Welcomes the user to the app with an introduction or CTA.
* Styling:
* Full-width background image with a semi-transparent overlay to make the text stand out.
* Large text for the main headline and a prominent call-to-action (CTA) button.
* Centered content with good padding and spacing.

css

/\* Hero Section Styling \*/

.hero {

background-image: url('hero-image.jpg'); background-size: cover; background-position: center; height: 400px; display: flex; justify-content: center; align-items: center; text-align: center; color: #ffffff; padding: 50px;

}

.hero h1 { font-size: 40px; margin-bottom: 20px;

}

.hero button { background-color: #ffcc00; color: #003366; padding: 15px 30px; border: none; cursor: pointer;

}

.hero button:hover { background-color: #003366; color: #ffffff;

}

## TESTING

Testing is a crucial aspect of the News Insights project to ensure that all components and features work as expected, maintain functionality during updates, and provide a reliable experience for users. The testing process helps identify bugs, optimize performance, and verify that the system behaves as expected across various devices and use cases.

**1. Unit Testing**

* Purpose: To test individual functions, methods, or components in isolation, ensuring that each part works correctly.
* Tools:
* Jest: JavaScript testing framework that allows for testing React components and functions.

**2. Integration Testing**

* Purpose: To test how different parts of the application work together (e.g., frontend and backend).
* Tools:
* Jest + Enzyme: For testing React components and simulating interactions between them

**3. Functional Testing**

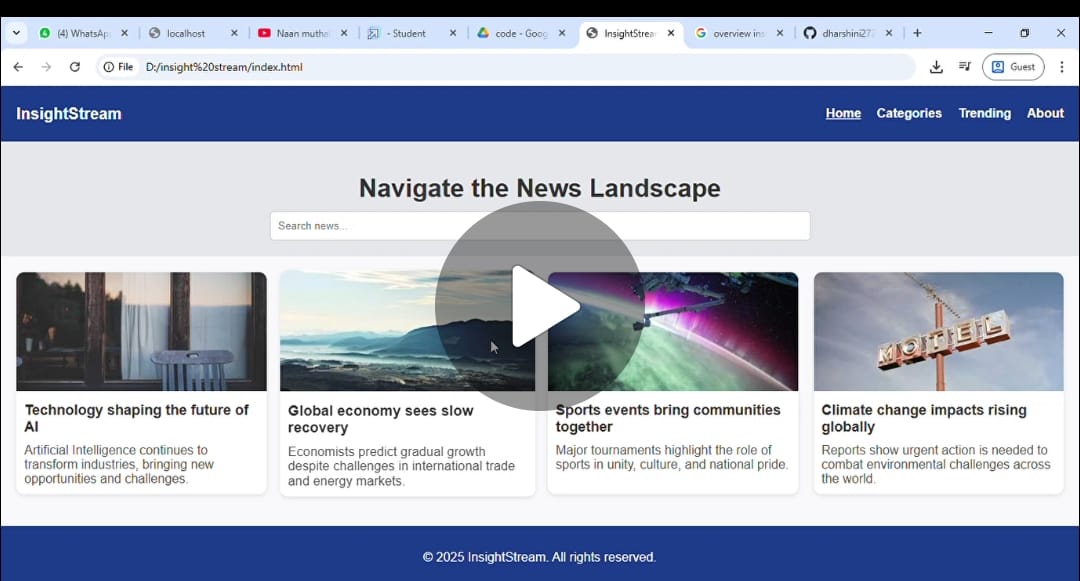
* Purpose: To validate the overall functionality of the application, ensuring that the system performs as expected from the user’s perspective.
* Tools:
* Cypress: For end-to-end (E2E) testing that mimics real user interactions.

Selenium: For automating browser interactions and checking functionality.

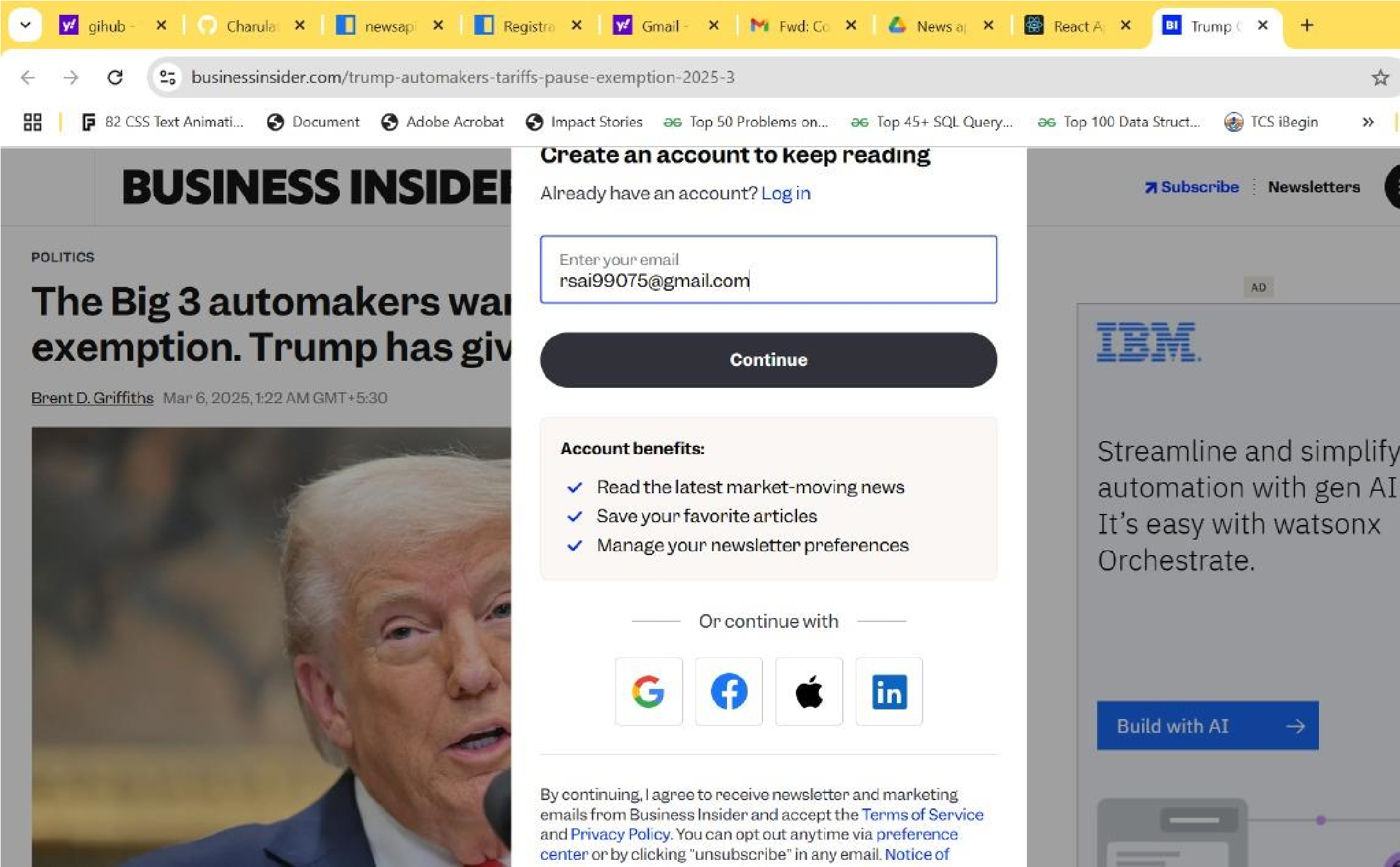
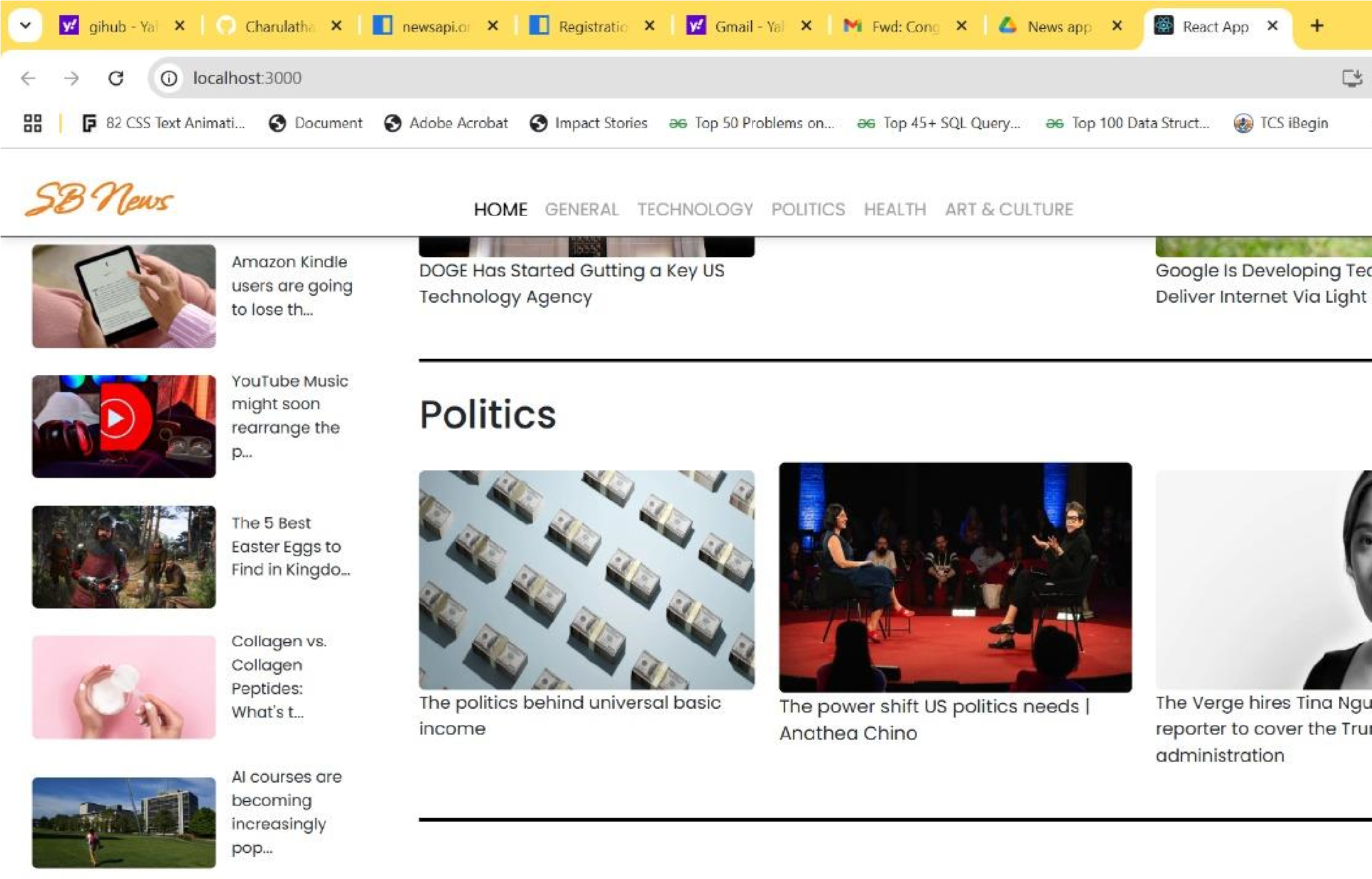
**4. UI Testing**

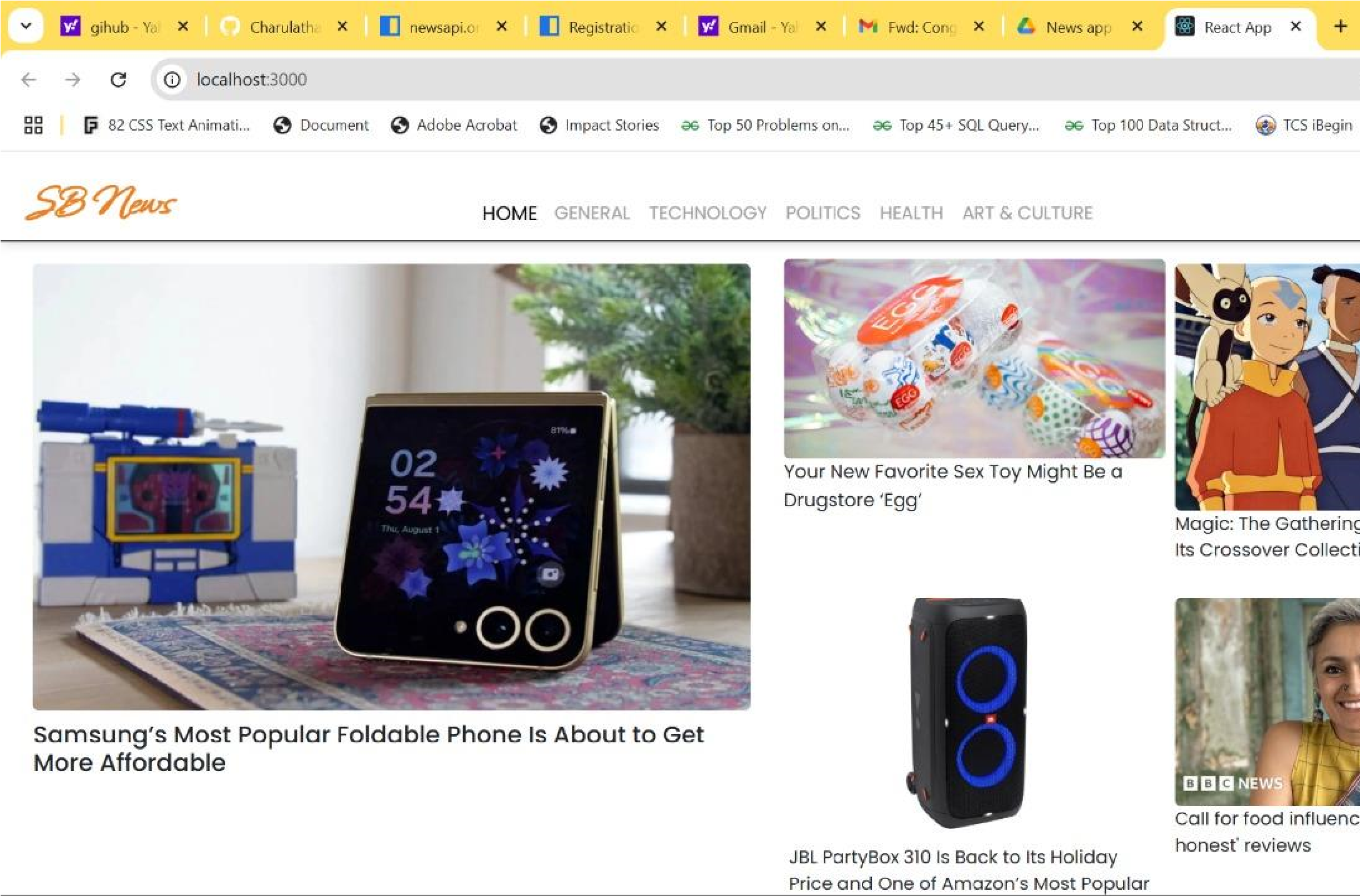
* Purpose: To verify that the user interface appears and functions as expected.
* Tools:
* Jest + React Testing Library: For testing individual components and ensuring they render correctly.
* Puppeteer: For browser-based UI testing.

SCREENSHOT AND DEMO



**DEMO LINK**: https://drive.google.com/drive/folders/1cGerjMnH-Pd3IGOYJVzGKycQf4IxLHB4





**THANKYOU**