**Project Documentation**

Insight Stream: Navigate the News Landscape

1. **Introduction**

* **Project Title**: Insight Stream: Navigate the News Landscape
* **Team ID –** NM2025TMID29983
* **Team Leader:** Thirumalini S
* [**202400700@sigc.edu**](mailto:202400700@sigc.edu)
* **Team Members:**
* Rajarajeshwari S : [202400059@sigc.edu](mailto:202400059@sigc.edu)
* Oviya T : [202400526@sigc.edu](mailto:202400526@sigc.edu)
* Sathiya Priya :[202400615@sigc.edu](mailto:202400615@sigc.edu)

1. **Project Overview**

* ***Description*:** Insight Stream is a news analytics and visualization platform that moves beyond a basic news feed. It aggregates articles from multiple sources and uses data analysis to help users understand news trends, media bias, and public sentiment. The platform provides a personalized and data-driven way to consume news, empowering users to be more informed and critical of the media they read.
* **Purpose:** The primary purpose is to combat information overload and media bias by providing a clear, insightful view of the news. Instead of just showing headlines, Insight stream helps users:
* **Discern trends:** Identify which topics are gaining or losing traction.
* **Recognize bias:** Compare how different news outlets report on the same story.
* **Understand sentiment:** See the overall emotional tone (positive, negative, or neutral) of news on a particular subject.

#### Features:

* **Real-time News Aggregation:** Gathers news from diverse sources to provide a comprehensive, up-to-the-minute feed.
* **Advanced Filtering & Search:** Allows users to find articles by keyword, category (e.g., politics, tech), or specific news source.
* **Sentiment Analysis:** Analyzes the emotional tone of articles to help users identify positive, negative, or neutral coverage.
* **Interactive Data Visualizations:** Presents complex information through charts and graphs, showing news trends and media bias in an easy-to-understand format.
* **Personalized Feeds:** Recommends articles and topics based on the user's interests and reading history.

1. Architecture

### Frontend: React.js with Bootstrap and Material UI

The frontend is the user-facing part of the application, responsible for everything the user sees and interacts with.

* **React.js:** This JavaScript library will be the core of the user interface. It allows for the creation of a **single-page application (SPA)**, which provides a fast and seamless user experience by only loading the necessary components. React's component-based architecture is perfect for building reusable UI elements like a news article card, a search bar, or a data visualization chart.
* **Bootstrap:** This is a CSS framework that provides pre-built, responsive components and a grid system. It will be used for laying out the application, ensuring it looks good and functions well on various screen sizes (desktop, tablet, and mobile). Bootstrap will handle the basic styling and responsive design.
* **Material UI:** This is another React component library based on Google's Material Design principles. It provides highly polished, modern-looking components like buttons, cards, and navigation bars. Material UI will be used to enhance the visual appeal and provide a consistent design language across the application, giving it a more professional and intuitive feel.

In summary, the frontend will be responsible for fetching data from the backend API, rendering the news feed and visualizations, handling user input (like search queries and filters), and managing the overall user interface.

* **Backend: Node.js and Express.js**

The backend is the server-side component that handles all the logic and data management behind the scenes.

* **Node.js:** This is a JavaScript runtime environment that allows you to run JavaScript on the server. Node.js is an excellent choice for a project like Insight stream because of its **non-blocking, event-driven I/O model**, which makes it highly efficient for handling a large number of concurrent connections—perfect for real-time data streaming and API requests.
* **Express.js:** This is a lightweight and flexible web application framework for Node.js. It will be used to build the **RESTful API endpoints** that the frontend will communicate with. Express.js simplifies tasks such as:
* **Routing:** Directing incoming HTTP requests (e.g., GET /api/news or POST /api/users).
* **Middleware:** Functions that can process requests before they reach the main handler (e.g., authentication checks).
* **API Management:** Creating endpoints to serve news data, handle user authentication, and manage user preferences.

The backend will be responsible for fetching news from external sources,processing the data (e.g., for sentiment analysis), and serving it to the frontend. It also handles all interactions with the database.

### Database: MongoDB

MongoDB is a **NoSQL, document-oriented database**. It will be used to store and manage the application's data.

* **Flexibility:** As a NoSQL database, MongoDB doesn't rely on a rigid, predefined schema. This is highly beneficial for a news application where the structure of news articles might vary between different sources. You can easily store complex, nested data structures within a single document.
* **Scalability:** MongoDB is known for its ability to scale horizontally, which is important for a growing application that will be handling a large volume of news data, user information, and analytical insights.
* **Data Storage:** The database will store various types of information:
* **User Data:** User profiles, login credentials, and saved preferences.
* **Project Information:** Details about the application itself, potentially for administrative purposes.
* **Applications:** This might refer to a separate module for job applications or other forms, but in the context of a news app, it could be for internal use or user-submitted content.
* **Chat Messages:** If a community or collaborative feature is planned, MongoDB can efficiently store conversation history.

1. **Setup Instructions**

The following instructions will guide you through setting up and running the **Insight stream** project. Ensure you have all the required software installed before proceeding.

### Prerequisites:

You need to have the following software installed on your machine to set up the project:

* **Node.js**: The JavaScript runtime environment. It includes npm (Node Package Manager), which is essential for installing project dependencies.
* **MongoDB**: The NoSQL database for data storage. You'll need to have a MongoDB server running locally or have access to a remote instance.
* **Git**: A version control system for cloning the project repository.
* **Visual Studio Code**: A powerful and popular code editor for developing the project.

While **React.js**, **Express.js**, and **Mongoose** are core technologies for the project, you don't need to install them globally. They are managed as dependencies within the project's package.json files and will be installed during the setup process.

* **Installation Steps:**

**Clone the Repository**: Open your terminal and clone the project from its Git repository.

* git clone [repository\_url]

Replace [repository\_url] with the actual URL of your project's repository.

**Install Client Dependencies**: Navigate into the client directory and install the required packages for the frontend.

* cd client

npm install

**Install Server Dependencies**: Go back to the root directory, then into the server directory, and install the backend packages.

* cd ../server

npm install

**Run the Project**:

* In your first terminal, start the backend server from the server directory:

npm start

* In a second terminal, start the frontend from the client directory:

npm start

The application will typically open in your browser at http://localhost:3000.

1. **Folder Structure**

The **Insight stream** project uses a clear, two-part folder structure to separate the frontend and backend code. This organization makes it easy to manage and scale the application.

* InsightStream/

|-- client/ # React Frontend Code

| |-- public/

| |-- src/

| | |-- components/ # Reusable UI components (e.g., ArticleCard.js)

| | |-- pages/ #Main application pages (e.g., HomePage.js, Dashboard.js)

| | |-- styles/ # CSS files or styling modules

| | |-- App.js # Main application component

| | |-- index.js # Entry point for the React app

| |-- package.json

|

|-- server/ # Node.js & Express.js Backend Code

| |-- config/ # Database and environment configurations

| |-- controllers/ # Business logic and request handlers (e.g., newsController.js) | |-- models/ # Mongoose schemas for MongoDB (e.g., Article.js, User.js)

| |-- routes/ # API endpoints (e.g., api/news, api/user)

| |-- services/ # External API interactions (e.g., fetching news from APIs)

| |-- middleware/ # Custom Express middleware (e.g., authMiddleware.js)

| |-- .env # Environment variables

| |-- server.js # Entry point for the server

| |-- package.json

|

|-- .gitignore # Specifies files to be ignored by Git

|-- package.json # Project-wide dependencies

|-- README.md # Project information and instructions

**6. Running the Application**

1. **Frontend:** cd client npm start
2. **Backend:** cd server npm start
3. **Access:** Visit <http://localhost:3000>

**7. API Documentation**

The **Insight stream** API is a simple, read-only RESTful service that provides access to news articles and their analysis. It's built with Node.js and Express.js, and all data is returned in **JSON** format.

### News API

This is the main API for fetching news content. All endpoints are public.

#### GET /api/news

* **Purpose**: Retrieves a list of the latest news articles. This endpoint is used to populate the main news feed.
* **Query Parameters**:
  + page: The page number for pagination.
  + limit: The number of articles per page.
* **Response**: A JSON array of news articles, each with a title, source, date, sentiment, and summary.

#### GET /api/news/:id

* **Purpose**: Fetches the complete details and analysis for a single article.
* **URL Parameter**:
  + id: The unique ID of the specific article.
* **Response**: A JSON object with the article's full content, sentiment score, and extracted topics.

### Search API

This endpoint allows users to search and filter the news feed.

#### GET /api/news/search

* **Purpose**: Searches for articles using various filters.
* **Query Parameters**:
* q: A keyword to search for.
* sources: A comma-separated list of news sources.
* topics: A comma-separated list of topics.
* **Response**: A JSON array of articles that match the search criteria.

**8. Authentication**

Authentication for this project will be handled using a modern, secure, and stateless method.

#### JWT-based Authentication

The primary method for user authentication is **JSON Web Tokens (JWTs)**. This approach is highly secure and scalable because it doesn't require the server to store session information for each user.

* **How it Works:**
  + - * **Login:** When a user logs in, the server verifies their credentials. If they are correct, the server creates a
* **JWT**—a small, digitally signed token that contains a unique user ID and other relevant information.
  + - * **Token Delivery:** The server sends this JWT back to the client (the user's browser).
      * **Future Requests:** For any future requests to a protected part of the application, the client sends this token along in the request's header.
      * **Verification:** The server receives the token and uses a secret key to verify its authenticity and ensure it hasn't been tampered with. If the token is valid, the request is processed; if not, it's rejected.

This process ensures that the user is who they claim to be without the server needing to continuously check a database for their session status.

#### 2. Middleware to Protect Private Routes

Middleware is a crucial Express.js feature that acts as a gatekeeper for your application's routes.

* **Function:** A specific middleware function, an authMiddleware, will be used to protect all private routes. It's a function that runs before the main route handler.
* **Protection:** Every time a user tries to access a protected route (like their profile page or saved articles), this middleware checks for a valid JWT in the request header.
* **Access Control:**
* **Valid Token:** If the JWT is present and valid, the middleware allows the request to proceed to its destination.
* **Invalid/Missing Token:** If the token is missing, expired, or invalid, the middleware immediately stops the request and sends a "401 Unauthorized" error back to the user.

This ensures that only authenticated and authorized users can access sensitive data or perform specific actions, providing a strong layer of security.

**9. User Interface**

The user interface for Insight stream is designed to be intuitive and data-rich, providing a seamless experience for exploring and understanding news. It is divided into key pages, each serving a specific purpose.

#### 1. Landing Page / Home Feed

This is the first page a user sees upon visiting the application. It provides an immediate and comprehensive overview of the current news landscape.

* **Header:**
* **Insight stream Logo:** A clickable logo that returns to the home page.
* **Search Bar:** A prominent search bar to find news by keyword, topic, or source.
* **Navigation Links:** Links to the main sections: "Home," "Dashboard," "About," and "Login/Profile."
* **Main Content:**
* **Featured Articles:** A carousel or a dedicated section highlighting the top trending news stories of the day.
* **Real-time News Feed:** A chronological feed of the latest news articles, with each article displayed as a card.
* **Article Card:**
* **Title and Source:** Clearly displays the article's headline and the name of the news outlet.
* **Image:** A thumbnail image to visually represent the article.
* **Sentiment Tag:** A small tag or icon indicating the article's sentiment (e.g., Green for positive, Red for negative, Yellow for neutral).
* **Short Snippet:** A brief summary of the article's content.

#### 2. Dashboard / Analytics Page

This is the core of Insight stream, where users can access data-driven insights and visualizations.

* **Header:** Same as the landing page, providing consistent navigation.
* **Topic Filters:** A list of clickable topics (e.g., Politics, Technology, Sports) to filter the displayed data.
* **Source Filters:** A list of news sources (e.g., BBC, CNN, Reuters) to analyze coverage from specific outlets.
* **Date Range Selector:** Allows users to view trends over a specific time period (e.g., past 24 hours, past week, custom range).
* **Main Content:**
* **Sentiment Breakdown Chart:** A pie chart or bar graph showing the distribution of positive, negative, and neutral articles for the selected topic or date range.
* **Topic Trends Graph:** A line graph that visualizes the prominence of a specific topic over time, showing when it gained or lost media coverage.
* **Source Comparison Panel:** A side-by-side view comparing how different news outlets report on the same story, including their sentiment scores and key phrases.

#### 3. Article Details Page

When a user clicks on an article card, they are taken to a dedicated page for that article.

* **Header:** The standard header for consistent navigation.
* **Article Content:**
* **Full Headline and Image:** The complete title and main image of the article.
* **Byline and Publication Date:** Author and the date of publication.
* **Full Text:** The complete content of the news article.
* **Insight stream Analysis Panel:**
* **Sentiment Score:** A prominent display of the article's sentiment score.
* **Extracted Keywords/Tags:** A list of key topics and entities mentioned in the article.
* **Related Articles:** A section that shows similar articles from different sources, helping the user compare perspectives.

#### 4. User Profile / Settings Page

This is where users can manage their account and personalize their experience.

* **Header:** Standard header.
* **Main Content:**
* **Account Information:** Allows users to view and update their username, email, and password.
* **Personalization Settings:** Users can select their preferred topics, news sources, and notification settings.
* **Saved Articles:** A list of articles the user has bookmarked for later reading.

**10. Testing**

### 1. Manual Testing

Manual testing involves a person directly interacting with the application to check its functionality and design.

* **Functionality**: We'll manually test all features to make sure they work as expected. This includes checking if the news feed updates, if filters return the correct articles, and if clicking on an article shows the right content and analysis.
* **User Interface (UI)**: We'll check the website's layout on different browsers and devices (like phones and tablets) to make sure it looks and works correctly.

### 2. Tools

We'll use specific tools to test the backend API and frontend behavior more effectively.

* **Postman**: This tool will be used to test the backend API. We'll send requests to our API endpoints (e.g., to get a list of news articles) to verify that they respond with the correct data and status codes.
* **Chrome DevTools**: We'll use these browser tools to check how the frontend and backend communicate. They'll help us monitor network requests, inspect the website's code, and check its responsiveness on different screen sizes.

**11. Screenshots**



**12. Demo link**

<https://drive.google.com/file/d/1K2EJ0imTMUfn-WodWz-PUNRCHOqa8Yxn/view?usp=drive_link>

**13. Future Enhancements**

### Future developments for Insight stream will focus on moving beyond basic aggregation to provide more advanced, personalized, and trustworthy news analysis.

* **Smarter Personalization:** Implement an AI-powered recommendation engine that learns from a user's reading habits to create a highly personalized news feed. This will go beyond simple keyword filters to suggest articles and topics the user is likely to find interesting.
* **Enhanced Media Bias Analysis:** Develop more sophisticated data visualizations to show how different news outlets cover the same story. This could include interactive charts that map out news sources based on their political leanings and highlight variations in sentiment and framing.
* **Real-time Trend Prediction:** Use predictive analytics to identify emerging news trends before they become main stream. The platform could alert users to a topic that is rapidly gaining traction in the media, giving them a head start on understanding a developing story.
* **Block chain for Trust:** Integrate block chain technology to create an immutable record of an article's provenance. This would allow users to verify the original source of an article, helping to combat misinformation and build trust in the news they are consuming.
* **Natural Language Querying:** Enable users to ask questions about the news in plain English (e.g., "What's the latest on climate change policy?") and receive a synthesized, data-driven answer, rather than just a list of articles.