**PROJECT DOCUMENTATION**

**INSIGHT** **STREAM**:**NAVIGATE** **AND** **NEWS** **LANDSCAPE**

**1.Introduction**:

1. **Project** **title**: Insight Stream:Navigate and News Landscape
2. **Team** **ID**: NM2025TMID37381
3. **Team** **Leader**:A.Yasmeen&yasmeen18042006@gmail.com
4. **Team** **Members**:

A.Abinaya & [abinaya90307@gmail.com](mailto:abinaya90307@gmail.com)

S.Vishali & [vvishalisantho@gmail.com](mailto:vvishalisantho@gmail.com)

M.Gayathri & [gayathrimurugan253@gmail.com](mailto:gayathrimurugan253@gmail.com)

**2.Project Overview**:

* **Purpose** :

SB Works connects clients and freelancers through project postings,bidding and real time communication

* **Features:**

1. Project Posting and Bidding – Clients can post projects with requirements, freelancers can submit bids.
2. Secure Chat System – Real-time messaging for smooth client-freelancer communication.
3. Feedback & Review System – Clients can review freelancers’ work, building credibility.
4. Admin Control Panel – Admins manage users, projects, disputes, and platform activity.

**3.Architecture**:

1. Data Layer (Collection)

News APIs (NewsAPI, GNews, Bing News API)

Web scraping (Python Scrapy / BeautifulSoup)

RSS Fees integration

2. Processing Layer (Pipeline)

Stream processing (Apache Kafka / RabbitMQ)

ETL (Extract, Transform, Load) pipelines

NLP (Named Entity Recognition, Sentiment Analysis, Topic Classification)

3. Storage Layer

Document Store (MongoDB / Elasticsearch for fast search)

Relational DB (PostgreSQL for structured metadata)

Data Lake (S3/HDFS for raw articles)

4. Insight Layer (AI/ML)

Trend analysis

Clustering similar news stories

Summarization models (e.g., Pegasus, LLM-based summarizers)

Bias detection (political leaning, sentiment polarity)

5. Application Layer (Frontend + API)

Web dashboard (React/Next.js + Tailwind)

API (FastAPI / Node.js)

Visualizations (charts for topic trends, maps for geo-news)

**4.Setup Instructions**:

Prerequisites:

Node.js (v18 or higher) and npm installed

Database (PostgreSQL / MySQL / MongoDB) running locally or via cloud service

Git installed for version control

Code editor (e.g., VS Code)

Environment variables configured:

JWT\_ACCESS\_SECRET and JWT\_REFRESH\_SECRET (long random strings)

DATABASE\_URL (connection string)

PORT (default: 4000)

* Installation steps:

1. System Requirements:

OS: Windows/Linux/macOS

Node.js (v14+)

MongoDB (v4+)

Modern browser (Chrome/Edge/Firefox)

2. Clone the Repository:

git clone https://github.com/username/insight-stream.git

cd insight-stream

3. Install Dependencies:

npm install

4. Configure Environment Variables:

Create a .env file:

PORT=5000

MONGO\_URI=<your\_mongodb\_connection\_string>

JWT\_SECRET=<your\_secret\_key>

API\_KEY=<your\_api\_key>

5. Run the Application:

npm run server # Start backend

npm run client # Start frontend (if separate)

6. Access the Application:

Open in browser: http://localhost:5000

7. Optional Tools:

Use Postman for API testing and Chrome DevTools for frontend debugging.

**5.Folder Structure**:

│── package.json

│── .env

│── src/

│ ├── server.js

│ ├── config/ # DB & JWT configs

│ ├── middleware/ # Auth middleware

│ ├── controllers/ # Route handlers

│ ├── routes/ # API routes

│ ├── models/ # DB models

│ └── services/ # Business logic

│

├── client/ # Frontend

│ ├── src/

│ │ ├── components/

│ │ └── pages/

│ └── package.json

│

└── docs/ # Setup, known issues, API docs

**6.Running the application**

* Frontend:

Cd client

npm start

* Backend:

Cd server npm

Start

* Access:Visit <http://localhost.3000>

**8.Authentication**:

* JWT-based authentication for secure login

JWT-based authentication secures login by issuing two tokens: a short-lived access token for API requests and a long-lived refresh token to get new access tokens.

Access tokens carry user identity in signed form and are verified on every request.

Refresh tokens are stored securely (e.g., HttpOnly cookies) and rotated to prevent misuse.

This approach improves security, scalability, and allows stateless session management.

* Middleware protects private routes

It checks for the Authorization: Bearer <token> header, verifies the JWT with the secret key, and if valid, attaches the user info to req.user.

If the token is missing or invalid, the middleware blocks access and returns 401 Unauthorized.

This ensures that only authenticated users can reach protected endpoints.

**9.User Interface**:

* Landing page:

A clean header with app name/logo.

A hero section with

short tagline (e.g., “Securely access your personalized dashboard”).

Two main buttons: Login and Sign Up.

Responsive layout with minimal colors (blue/white/grey works well).

Optional: background illustration or gradient for a modern look.

* Free lancer dashboard:

The Freelancer Dashboard provides a space for freelance journalists and writers to contribute content, manage assignments, and track their performance. It includes:

Content Submission: Upload and edit articles or reports.

Assignments: View and apply for trending topics.

Analytics: Track views, ratings, and engagement.

Earnings: Monitor income and withdrawals.

Profile: Maintain a portfolio and credibility score.

* Admin Panel:

Admin Login (JWT auth).

User Management → View, edit, ban users.

News Moderation → Approve/remove spam/fake news.

Analytics Dashboard →

Ingestion stats (#articles/day).

Source distribution (pie chart).

Sentiment analysis chart.

* Project Details:

Header/Navbar → App name, user profile, logout button.

Project Title & Description → Clear display of the project name, summary, and goals.

Project Metadata → Owner, creation date, last updated, status (active/completed).

Tabs or Sections → Tasks, Team Members, Files/Attachments, Activity/Timeline.

Actions → Edit project, add members, upload documents, or delete project.

**10.Testing**:

* Manual testing during milestones:

1. Define milestone scope → List which modules/features must be completed by that milestone.

2. Create test cases/checklists → Based on requirements, write simple manual steps to verify functionality (input, output, error handling).

3. Execute tests → Run the app manually, record pass/fail results for each case.

4. Log defects → Note any bugs, unexpected behaviors, or missing features.

5. Re-test after fixes → Ensure issues are resolved before signing off the milestone.

* Tools: Postman,Chrome Dev Tools

Postman

Postman is used to test and validate APIs of Insight Stream: Navigate the News Landscape. It helps developers check endpoints like user login, profile, preferences, and freelancer submissions. With Postman, we can send requests, verify responses, handle authentication tokens, and ensure the backend works correctly.

Chrome DevTools

Chrome DevTools is used to debug and test the frontend. It allows inspection of UI elements, monitoring network calls, checking API requests/responses, viewing storage (tokens, preferences), and testing performance. It ensures that dashboards (user and freelancer) run smoothly across devices.

**11.Screenshots or demo:**

[**https://github.com/abinaya2947/InsightStream.git**](https://github.com/abinaya2947/InsightStream.git)

**12.Known Issues:**

1. Authentication

Refresh token sometimes not rotated properly, leading to forced logout.

JWT expiry messages are unclear to end-users.

2. Navigation / UI

Slow response when switching between “News Sources” and “Trends” tabs.

Inconsistent styling on mobile (cards overlap in smaller screens).

3. News Data / API

Some sources fail to load due to API rate limits.

Duplicate articles appear when filtering by region.

4. Search & Filtering

Keyword search is case-sensitive (should be case-insensitive).

Filter resets when user refreshes the page.

5. Performance / Stability

Dashboard load time exceeds 5s when >1000 articles are fetched.

Occasional API timeout errors during peak usage.

**13.Future Enhancements:**

1. AI-Powered Recommendations: Personalized news suggestions based on user interests, reading history, and trending topics.

2. Advanced Analytics: Insights and visualizations for users and freelancers on engagement trends, sentiment analysis, and topic patterns.

3. Multilingual Support: News content and interface available in multiple languages to reach a wider audience.

4. Mobile App Development: Dedicated Android/iOS apps for better accessibility and push notifications.

5. Enhanced Freelancer Tools: Improved content management, collaboration features, and performance-based rewards.

6. Integration with Social Media: Sharing, tracking, and trend analysis from platforms like Twitter, LinkedIn, etc.

7. Offline Mode: Save articles for offline reading with automatic synchronization.

8. Fact-Checking AI: Automatic verification of news credibility to reduce misinformation.