

INSIGHTSTREAM: NAVIGATE THE NEWS LANDSCAPE

1. INTRODUCTION :

* Project Title : InsightStream: Navigate the News Landscape
* Team ID : NM2025TMID30503
* Team Leader : Matheswaran.S (maddymass6@gmail.com)
* Team Members: Deepak.L (dl8700092@gmail.com)

Dharmalingam.R(dharmavelu204@gmail.com)

Dineshkumar.S (kdinesh76652@gmail.com)

Gokul.S (gokulvino940@gmail.com)

1. PROJECT OVERVIEW:

* Purpose: The purpose of the Insight Stream: Navigate the News Landscape project is to help people critically engage with the overwhelming flow of news and information in today’s media environment.

Here’s a clear breakdown of its purpose:

1. Develop Critical Thinking Skills

Teach readers/viewers how to identify bias, misinformation, and unreliable sources. Encourage questioning and analysis rather than passive consumption.

2. Improve Media Literacy

Provide tools to distinguish between fact, opinion, and propaganda. Help users understand how headlines, images, and framing influence perception.

3. Encourage Responsible News Consumption

Guide audiences to balance perspectives by checking multiple sources. Promote awareness of echo chambers and algorithm-driven content.

4. Empower Informed Decision-Making

Ensure people can form opinions based on credible information. Support civic engagement, discussions, and participation in democracy.

5. Build Lifelong News Navigation Habits

Instill habits of cross-checking, verifying facts, and recognizing trustworthy outlets. Encourage sustainable, healthy ways to interact with daily news without feeling overwhelmed.

* Features:

1. News Source Analysis

Identifies credible vs. unreliable sources. Explains how ownership, funding, and bias affect reporting.

2. Fact-Checking Tools

Teaches how to verify information using trusted platforms.Provides step-by-step methods to spot fake or misleading news.

3. Bias Detection

Helps recognize language framing, selective reporting, and emotional tone. Encourages comparing multiple perspectives on the same event.

4. Interactive Learning Modules

Quizzes, case studies, and real-world examples. Practical exercises in analyzing headlines, articles, and media posts.

5. Digital Media Literacy

Explains how algorithms and social media shape what news we see. Guides users in escaping echo chambers.

6. Critical Thinking Framework

Encourages asking “Who wrote this? Why? What’s missing?” Promotes reflective, independent judgment instead of blind trust.

7. Accessible & User-Friendly Format

Designed for students, educators, and general audiences. Can be adapted into workshops, assignments, or self-study guides.

8. Real-World Application

Equips users to apply skills when reading daily news. Supports informed civic engagement and decision-making.

1. ARCHITECTURE:

* Frontend:

Technology Stack:

React.js (or Angular/Vue) for building an interactive and responsive interface. Tailwind CSS / Bootstrap for styling and layout.

* Backend:

Technology Stack:

• Node.js with Express.js (or Python Flask/Django) – handles server logic and APIs.

• Database: MongoDB / MySQL to store user data, quiz results, and news analysis content.

• Authentication: JWT (JSON Web Tokens) for secure login and session handling.

* Database:

Database Choice:

• MongoDB (NoSQL) – for flexible, document-based storage of articles, user progress, and learning modules.

• (Alternative: MySQL/PostgreSQL if relational structure is preferred).

Database Functions:

\* Securely stores user data and progress.

\* Provides quick access to news content for bias/fact-check analysis.

\* Supports admin tools for updating modules and reviewing feedback.

\* Ensures scalability to handle large datasets of news articles.

1. SETUP INSTRUCTIONS:

* Prerequisites:

1. Software Requirements

Node.js (for backend server and package management)

Express.js / Flask / Django (depending on chosen backend framework)

MongoDB / MySQL (for database management)

React.js / Angular / Vue.js (for frontend development)

Git (for version control and collaboration)

Visual Studio Code (VS Code) or any suitable IDE

2. System Requirements

Operating System: Windows / Linux / macOS

Minimum RAM: 4 GB (8 GB recommended)

Processor: Intel i3 / AMD equivalent (i5 or higher recommended)

Storage: At least 500 MB free space for dependencies and database

3. Additional Dependencies

npm (Node Package Manager) for installing frontend & backend libraries

Browser (Chrome/Firefox/Edge) for running and testing frontend modules

Postman / Thunder Client for testing API endpoints

* Installation Steps:

1. Clone the Repository

git clone <repository\_url>

cd insightstream

2. Frontend Setup

cd client

npm install # Install frontend dependencies

npm start # Start the frontend development server

3. Backend Setup

cd ../server

npm install # Install backend dependencies

npm start # Start the backend server

4. Database Setup

Install and start MongoDB (or MySQL, if chosen).Create a database named insightstream\_db.Configure connection details inside server/config/db.js (for MongoDB) or .env file.

5. Access the Application

Open your browser and visit:

http://localhost:3000

Login/Register to explore features.

1. FOLDER STRUCTURE:

│

├── client/ # Frontend (React.js or chosen framework)

│ ├── public/ # Static files (HTML, images, icons)

│ ├── src/ # Source code

│ │ ├── components/ # Reusable UI components (Navbar, Footer, Cards)

│ │ ├── pages/ # Main pages (Dashboard, News Analysis, Quizzes)

│ │ ├── assets/ # Images, styles, fonts

│ │ ├── services/ # API calls to backend

│ │ └── App.js # Root React component

│ └── package.json # Frontend dependencies

│

├── server/ # Backend (Node.js / Express.js or Python)

│ ├── routes/ # API route definitions (users, news, quizzes)

│ ├── models/ # Database models (User, Article, Quiz, Reports)

│ ├── controllers/ # Handles logic for each route

│ ├── config/ # Database connection & environment variables

│ └── server.js # Entry point for backend

│

├── database/ # Database scripts and seed data

│ └── schema.sql/json # Schema or initial data (if needed)

│

├── docs/ # Project documentation (purpose, features, ERD)

│

├── .env # Environment variables (DB credentials, API keys)

├── README.md # Project overview & instructions

└── package.json # Root dependencies (if monorepo setup)

1. RUNNING THE APPLICATION:

* Frontend:

cd client

npm start

* Backend:

cd server

npm start

* Access:

<http://localhost:3000>

1. API DOCUMENTATION:

* User:

POST /api/user/register → Register a new user

POST /api/user/login → Authenticate user & return JWT token

GET /api/user/profile/:id → Fetch user profile & progress

PUT /api/user/update/:id → Update user details

DELETE /api/user/:id → Delete a user account

* Project:

POST /api/projects/create → Create a new learning project or analysis task

GET /api/projects/:id → Fetch details of a specific project

GET /api/projects → Fetch all available projects/modules

PUT /api/projects/:id → Update a project or module

DELETE /api/projects/:id → Remove a project

* Chats:

POST /api/chat/send → Send a message to another user

GET /api/chat/:userId → Fetch chat history with a specific user

GET /api/chat/conversations/:userId → Fetch all conversations for a user

DELETE /api/chat/:chatId → Delete a specific chat message

1. AUTHENTICATION:

Method Used:

• JWT (JSON Web Token) – for secure user authentication and session handling.

Flow:

1. User Registration/Login

When a user registers or logs in, credentials are verified against the database. On success, the server generates a JWT token containing user ID and role.

2. Token Storage

The JWT token is sent to the frontend and stored in localStorage / sessionStorage.

3. Protected Routes

Every request to protected APIs (e.g., /api/news/analyze, /api/projects/create) must include the JWT token in the request header:

Authorization: Bearer <token>

4. Token Verification

The backend verifies the token using a secret key.

If valid → request proceeds.

If invalid/expired → request is denied with 401 Unauthorized.

Security Features:

• Encrypted passwords using bcrypt before storing in database.

• Token expiration (e.g., 1 hour) to reduce misuse risks.

• Role-based access:

User: can analyze news, take quizzes, chat.

Admin: can manage projects, modules, and monitor reports.

1. USER INTERFACE:

The UI of the project is designed to be clean, interactive, and user-friendly, ensuring easy navigation for all users.

Landing Page

• Introduces the project and its purpose

• Quick links to Register/Login

• Highlights of features (News Analysis, Fact-Checking, Quizzes)

User Dashboard

• Personalized view after login

• Displays user progress, recent activities, and quiz results

• Quick access to News Analysis, Modules, and Reports

News Analysis Page

• Input box to paste a news link or article text

• Displays analysis results: bias score, credibility rating, fact-check status

• Comparison with multiple sources

Learning Modules & Quizzes

• Interactive lessons on media literacy

• Multiple-choice quizzes with instant feedback

• Gamified progress tracking (badges, scores)

Fact-Checking Page

• Search bar for claims/articles to be verified

• Integration with third-party fact-check APIs

• Displays results as True / False / Mixed

Chat System

• Real-time messaging between users and mentors/educators

• Conversation history stored securely

• Notifications for new messages

Reports & Insights Page

• Graphical representation of user progress (charts, stats)

• Downloadable reports for educators and admins

Admin Panel (for Admin Users)

• Manage modules, quizzes, and fact-check sources

• Monitor user activity and feedback

• Control access and update content

1. TESTING:

Testing Approach:

• Manual testing during development milestones.

• API testing with Postman / Thunder Client.

• Browser-based testing for frontend modules (Chrome Dev Tools).

Types of Testing Performed:

1. Unit Testing

Tested individual functions (e.g., login validation, quiz scoring). Backend routes tested for expected JSON responses.

2. Integration Testing

Verified communication between frontend and backend.

Example: Submitting a news article in frontend correctly triggers backend API and shows analysis results.

3. Functional Testing

Checked all features against requirements:

User Registration/Login

News Analysis

Fact-Checking

Quizzes & Reports

Chat Messaging

4. UI/UX Testing

Verified responsiveness across devices (desktop, tablet, mobile). Ensured accessibility (readable fonts, color contrasts, alt-texts).

5. Security Testing

Password encryption verified with bcrypt. JWT authentication tested for protected routes. Invalid tokens rejected with 401 Unauthorized.

Testing Tools Used:

• Postman → API request/response validation

• Chrome Dev Tools → UI and performance testing

• MongoDB Compass / MySQL Workbench → Database validation

• Jest / Mocha (optional) → Automated unit tests for backend

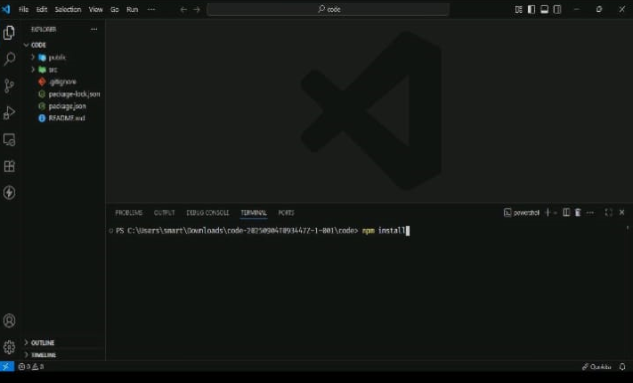
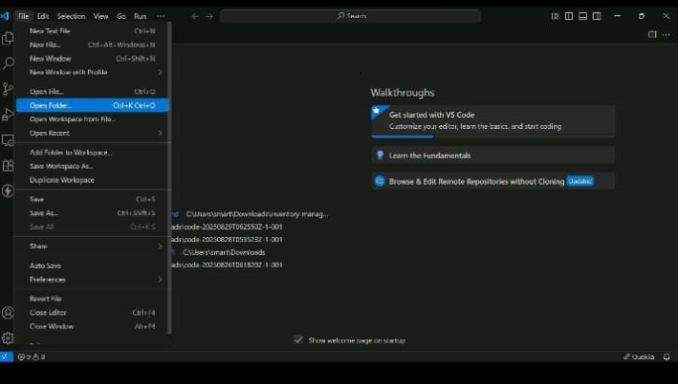
Outcome:

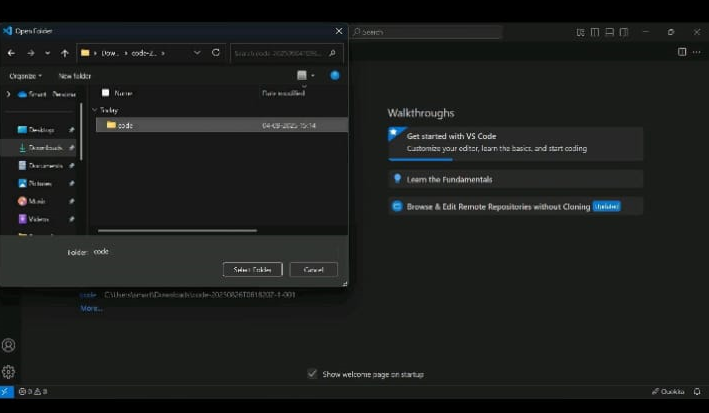
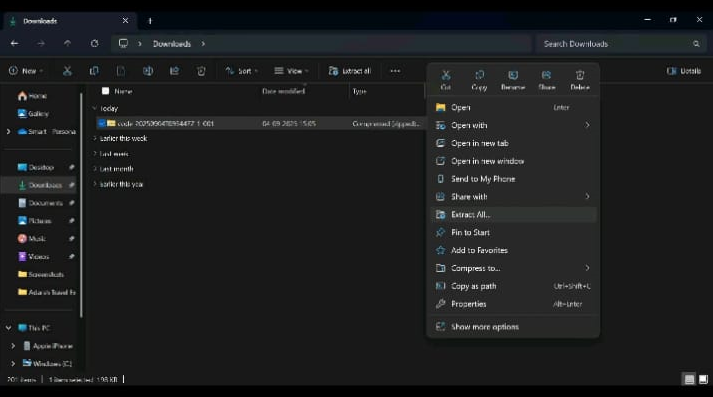
• All core modules (User, News Analysis, Fact-Checking, Quiz, Reports, Chat) passed functionality checks.

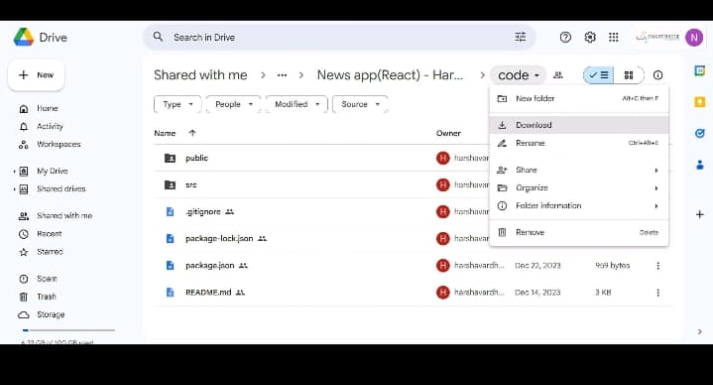
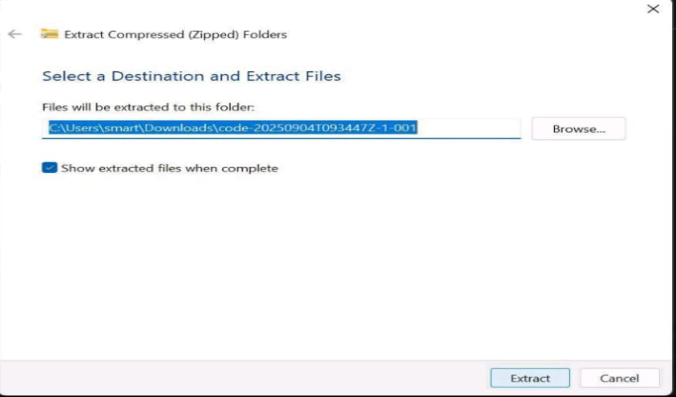
• Minor UI bugs identified and fixed (alignment, form validations).

• System is stable for end-user testing and deployment.

1. SCREENSHOTS OR DEMO:





1. KNOWN ISSUES:

1. Limited Fact-Check Sources

Current version depends on a few third-party APIs. Some claims may return “No Data Found.”

2. Basic Bias Detection

Bias analysis relies on keyword-based and sentiment analysis. Contextual bias or deeper propaganda techniques may not always be detected.

3. Real-Time Chat Limitations

Chat module supports text-only communication. No support yet for media

1. FUTURE ENHANCEMENT:

To improve and expand the project in the future, the following enhancements are planned:

1. AI-Powered News Verification

Integrate advanced AI/ML models to automatically detect misinformation, bias, and fake news in real-time.

2. Multi-Language Support

Provide content, quizzes, and analysis in multiple languages to reach a wider audience.

3. Mobile Application

Develop Android and iOS apps for easier accessibility and learning on the go.

4. Gamification Features

Add badges, points, and leaderboards to make learning more engaging.

5. Advanced Reporting System

Generate detailed reports with insights for educators, policymakers, and researchers.

6. Collaboration & Community Features

Create discussion forums or group chats where users can analyze and debate news together.

7. Integration with External Fact-Check Databases

Real-time API integration with global fact-checking organizations (e.g., PolitiFact, FactCheck.org).

8. Personalized Learning Path

Use AI to recommend modules and quizzes based on user performance and interests.