**INSIDE NEWSLANDSCAPE**

**1. Introduction :**

**Project Title: Newslandscape**

**Team ID:**[NM2025TMID37623]

**Team leader:**

[Priya dharshini.S& priyadharshinisasikumar17@gmail.com]

**Team Members:**

[Sridevi.A.P & apsridevi06@gmail.com]

[Monisha.B & monilogi14@gmail.com]

[Vidhya.M & mvidhya292@gmail.com]

**2.Project overview:**

The Inside News Landscape project explores the dynamic and ever-evolving world of modern journalism, focusing on how information is gathered, verified, and delivered in today’s digital age. This project provides an in-depth analysis of news production, distribution channels, and audience engagement across multiple platforms, highlighting the influence of technology, social media, and real-time reporting. By examining current trends, challenges, and opportunities within the news ecosystem, the initiative aims to present a clear understanding of how stories are shaped, how public perception is influenced, and how credible journalism can thrive amidst misinformation. Through research, case studies, and visual storytelling, the project offers valuable insights into the shifting landscape of news and its critical role in shaping society.

• Features:

* **Real-Time Updates** – Delivers the latest news and developments as they happen.
* **Category Segmentation** – News organized into sections such as politics, business, sports, entertainment, and technology.
* **Interactive Interface** – User-friendly design with easy navigation and quick access to trending stories.
* **Multimedia Content** – Supports images, videos, and infographics to enhance storytelling.

**3.Architecture:**

The architecture of the Inside News Landscape platform is designed as a multi-tier system to ensure seamless news delivery and user engagement. At the core, a data ingestion layer collects and aggregates information from various trusted sources, APIs, and live feeds. This is followed by a processing and analytics layer, where news data is filtered, verified, and categorized using AI-powered algorithms for relevance and accuracy. A content management system (CMS) enables editors to manage articles, multimedia content, and updates efficiently. The application layer hosts the user interface, offering a responsive web and mobile experience with personalized news feeds and interactive features. A robust database layer stores articles, user profiles, and historical data, while a cloud infrastructure ensures scalability, high availability, and real-time updates. Security measures, including encryption and access control, are embedded across all layers to protect sensitive information and maintain data integrity.

* **Frontend:** Built with **React.js**, the user interface is developed using **Bootstrap** and **Material UI** for a responsive, visually appealing, and interactive design. This layer manages real-time updates, personalized news feeds, and seamless navigation for users across web and mobile devices.
* **Backend:** Powered by **Node.js** and **Express.js**, the server handles core logic, API endpoints, and secure communication between the client and the database. It also manages user authentication, content retrieval, and data processing.
* **Database:** **MongoDB** serves as the primary database, storing user data, project details, applications, chat messages, and real-time news content. Its NoSQL structure ensures flexibility and scalability for handling large volumes of dynamic news updates.

**4. Setup Instructions :**

Follow the steps below to set up and run the **Inside NewLandscape** project on your local machine.

**Prerequisites**

Ensure the following software/tools are installed on your system:

* **Node.js** – For running the backend server.
* **MongoDB** – For database storage.
* **Git** – For cloning and version control.
* **React.js** – For frontend development.
* **Express.js** – For backend server logic.
* **Mongoose** – For MongoDB object data modeling.
* **Visual Studio Code (VS Code)** – Recommended code editor.

**Installation Steps :**

Follow these steps to set up the project on your local machine:

1. Clone the Repository
2. git clone <repository-url>
3. cd inside-newlandscape
4. Install Client (Frontend) Dependencies
5. cd client
6. npm install
7. Install Server (Backend) Dependencies
8. cd ../server
9. npm install
10. **Start the Application**
    * **Client (Frontend):**
    * npm start

Runs on [**http://localhost:3000**](http://localhost:3000/)

* + **Server (Backend):**
  + npm run dev

Runs on [**http://localhost:5000**](http://localhost:5000/) (or your configured port)

**5. Folder Structure:**

├─ client/ # React frontend

│ ├─ public/ # Static files (index.html, favicon, etc.)

│ └─ src/

│ ├─ components/ # Reusable UI components

│ ├─ pages/ # Page-level React components

│ ├─ assets/ # Images, icons, and styles

│ ├─ services/ # API calls & client-side utilities

│ └─ App.js # Main React application entry

│

├─ server/ # Node.js backend

│ ├─ routes/ # Express route definitions

│ ├─ models/ # Mongoose schemas & Mong

│ ├─ controllers/ # API logic and request handling

│ ├─ config/ # Database connection & environment setup

│ ├─ middleware/ # Authentication, error handling, etc.

│ └─ server.js # Backend entry point

│

├─ .env # Environment variables (Mongo URI, Ports)

├─ package.json # Project metadata & dependencies

└─ README.md # Project documentation

* client/ holds the **React.js frontend** (using Bootstrap + Material UI).
* server/ contains the **Node.js + Express backend**, MongoDB models, and API endpoints.
* config/ typically stores database configuration (like db.js).
* .env should contain sensitive information (e.g., MONGO\_URI, PORT).

**6. Running the Application:**

Follow these commands to launch both the **frontend** and **backend**.

* **Frontend: http://localhost:3000**
* **Backend/API Base: http://localhost:5000**

**7. API Documentation :**

**Base URL:** http://localhost:5000/api

**User**

* **POST** /user/register – Register a new user.
* **POST** /user/login – User login & token generation.

**Projects**

* **POST** /projects/create – Create a new project/news item.
* **GET** /projects/:id – Get details of a specific project.

**Applications**

* **POST** /apply – Submit an application for a project.

**Chats**

* **POST** /chat/send – Send a chat message.
* **GET** /chat/:userId – Fetch chats for a user.

**8. Authentication :**

* **JWT-Based Authentication:** Uses **JSON Web Tokens (JWT)** to provide secure user login and session management.
* **Protected Routes:** Backend **middleware** verifies the JWT token to restrict access to private routes such as project creation, applications, and chats.
* **Header Usage:** Clients must send the token in the request header:
* Authorization: Bearer <token>

This ensures only authenticated users can access sensitive APIs**.**

**9. User Interface – Inside NewsLandscape**

* **Landing Page** – Welcomes users with the latest news highlights, featured projects, and easy navigation options.
* **Freelancer Dashboard –** Allows freelancers to manage their profiles, browse projects/news, apply for opportunities, and track applications.
* **Admin Panel –** Enables administrators to manage users, approve or remove projects/news, monitor chats, and maintain platform security.
* **Project Details Page –** Displays complete project or news information, including descriptions, categories, contributors, and application options.

**10. Testing :**

* **Manual Testing:** Performed at key development milestones to verify core functionality, UI responsiveness, and user flows.
* **Tools Used:**
  + **Postman** – For testing API endpoints and verifying request/response data.
  + **Chrome DevTools** – For debugging frontend components, checking network activity, and monitoring performance.

**12. Known Issues :**

* **Limited Error Handling:** Some API endpoints may return generic error messages instead of detailed feedback.
* **Performance on Large Data Sets:** Rendering large volumes of news/projects can cause minor UI lag.
* **Authentication Timeout:** JWT token expiry handling may require a more seamless refresh mechanism.
* **Cross-Browser Compatibility:** Minor styling inconsistencies may occur on older browsers.

**13.Future Enhancements :**

* **Real-Time News Updates:** Integrate live news feeds and push notifications for instant updates.
* **AI-Powered Recommendations:** Use machine learning to deliver personalized news and project suggestions.
* **Mobile App Version:** Develop Android and iOS apps for better accessibility and user engagement.
* **Advanced Search & Filters:** Add keyword-based search, trending topics, and category filters for improved navigation.
* **Multilingual Support:** Provide content in multiple languages to reach a wider audience.
* **Offline Mode:** Enable cached news reading when users are offline.
* **Role-Based Access Control:** Enhance admin and freelancer permissions for better security and management.