# Insight stream-news landscape

## 1. Introduction

**Project Title: InsightStream**

**Team Members:**

**Team leader:**

THAMILARASI S -tamil2006tkm@gmail.com

**Team mates:**

LIVANI G- livanigopi06@gmail.com

PRAVEENA P - praveenaarthipraveena@gmail.com

SURUTHI G - Suruthig401@gmail.com

## 2. Project Overview

**Purpose:**

The purpose of Insight Stream (in general business/learning/technology contexts) is to provide a continuousflow of meaningful observations, analysis, or feedback that helps inunderstanding, decision-making, and improvement.

**Features:**

1. Continuous Flow of Insights – Provides real-time or regular updates instead of one-time reports.

2. Data Integration – Connects with multiple sources (databases, apps, tools) to pull information together.

3. Visualization Tools – Uses dashboards, graphs, and charts to make insights easier to understand.

4. Customization – Users can filter, sort, and personalize insightsbased on their needs.

## 3. Architecture

### 🏗️ Component Structure (Tree Diagram)

```
App
 ├── Navbar
 ├── Routes
 │    ├── Home (shows top headlines)
 │    │     └── NewsList
 │    │           └── NewsCard
 │    │
 │    ├── Category (news by category)
 │    │     └── NewsList
 │    │           └── NewsCard
 │    │
 │    ├── SearchResults
 │    │     └── NewsList
 │    │           └── NewsCard
 │    │
 │    └── About
 │
 └── Footer
```

---

### 🔄 State Management (Context API)

- Managed states:
  - `articles` → fetched news articles
  - `loading` → true/false while fetching
  - `error` → API error handling
  - `searchQuery` → stores user input for searching

👉 Using Context API avoided **prop drilling** (passing props manually down many components).

---

### 🛣️ Routing (React Router)

- Implemented navigation with `react-router-dom`.
- **Routes Defined:**
  - `/` → Home page (top headlines)
  - `/category/:name` → News by category (e.g. Sports, Tech)
  - `/search/:query` → Search results page
  - `/about` → About app

👉 **Navbar** provides links for easy navigation.

## 4.Setup Instructions for Our News Application

### 1. Install Required Tools

- Install **Node.js** (comes with npm).
- Install **Visual Studio Code (VS Code)** as the code editor.
- Install **Git** (optional, if code is from GitHub/CodeLink).

---

### 2. Get the Project Code

- Download the project files (from SmartInternz CodeLink).
- Extract the zip file into a project folder.
- Open the folder in **VS Code**.

---

### 3. Install Dependencies

Open the terminal inside VS Code and run:

```
npm install
```

👉 This installs all the required packages (like react, react-router-dom, etc.) listed in **package.json**.

---

### 4. Add News API Key

- Go to https://newsapi.org and create an account.
- Copy your **API key**.
- Open the project code and replace the placeholder with your API key (usually in `config.js` or directly in fetch function).

---

## 5. Folder Structure

- src/components/ → Reusable UI elements (Navbar, Card, Footer).
- src/pages/ → Page-level views (Home, About, NewsDetails).
- src/assets/ → Images, icons, fonts, and stylesheets.
- src/hooks/ → Custom hooks (e.g., useFetchNews).

- src/utils/ → Helper functions (e.g., date formatting, API calls).
- src/App.js → Root application logic.
- src/index.js → Entry point for rendering React app.

## 6. Run the Application

- Start the development server by running: “npm start”
- The app will run at http://localhost:3000/.
- It will automatically open in the browser.

## 7. Component Documentation

Key Components:

- NewsCard → Displays article title, image, and description.
- Navbar → Provides navigation between pages.
- Footer → Shows copyright & contact info.

Reusable Components:

- Button, Loader/Spinner, Modal for alerts

## 8. State Management

**Global State:**

Managed using Context API (stores API responses, theme state).

**Local State:**

Managed with useState for forms, search filters, toggles.

**Example:**

Search bar input state maintained locally, fetched news stored globally.

## 9. User Interface

1. Clean and simple UI.
2. Responsive across devices (mobile, tablet, desktop).

**10.Styling**

We styled the project using HTML structure (via JSX) + CSS. Each component has its own CSS rules for layout and design. Flexbox/Grid made the app responsive, while hover effects and consistent themes improved the overall user experience.

**11.Testing**

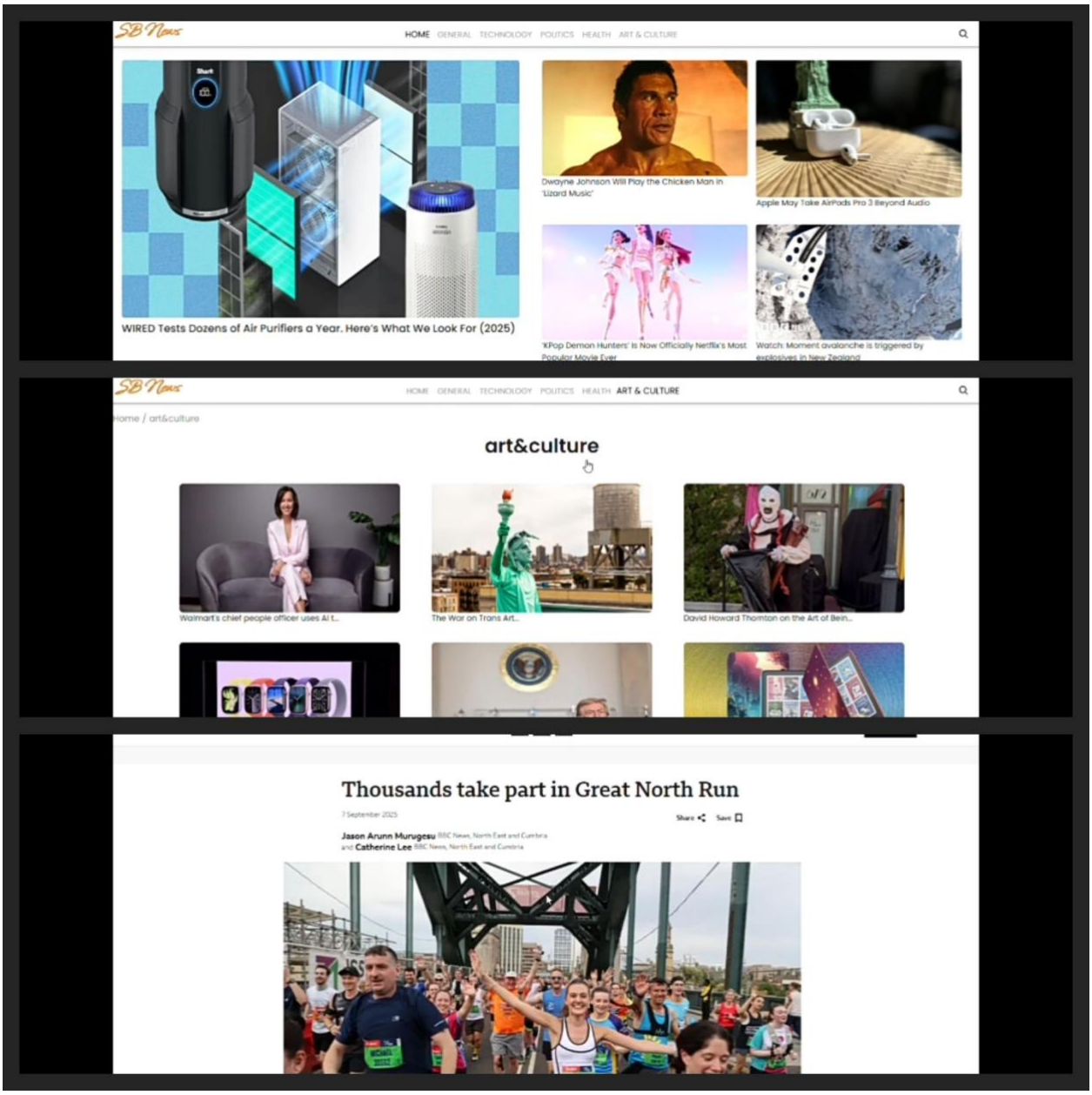
We tested the app by checking functionality, navigation, error handling, responsiveness, and performance manually. The application worked smoothly across different devices and browsers.

## 12.Screenshots and demo

Demo video link:

https://drive.google.com/file/d/1kKX7_H4NpvNH_iOn2fkqBKruklgYhcUX/view?usp=sharing

Screen shots:




## 13.Known issues

1. Notification Overload – Too many alerts, even for small updates.
2. Slow Loading Times – lag when loading videos or images.
3. Advertisements – Pop-ups or banner ads can interrupt reading.

## 14.Future enhancements

1.AI/ML Capabilities – Uses artificial intelligence to detect patterns, anomalies, and predict outcomes.
2. Action-Oriented – Focuses not just on showing data but also on suggesting or enabling next steps.
3. Feedback Loop – Allows continuous refinement of decisions and strategies based on new insights.