



Dharmamurthi Rao Bahadur Calavala CunnanChetty's Hindu College

Pattabiram, Chennai-55
College Code: **UNM0110**



Submitted to



InsightStream Navigate the News Landscape

**BACHELOR OF COMPUTER APPLICATIONS
SUBMITTED BY**

Team Members (Team ID: 105673)

Name	Email id
Santhosh Raj V (team Leader)	sksanthosh88409@gmail.com
Prashanth N	prashanthnp9894@gmail.com
Magesh Kumar T	maheshkumart542@gmail.com
Saran P	bathamanapan@gmail.com
Kamal Kannan M.M	kamalakannan22992@gmail.com

Table of content

Section Number	Section Title
1	Introduction
2	Project Overview
3	Features
4	Technical Architecture
5	Installation Guide
6	Configuration
7	Usage Instructions
8	API Integration
9	Source Code Explanation
9.1	HTML
9.2	Java Script
9.3	CSS
10	Screen layout
11	Future Development
12	conclusion

1. Introduction

InsightStream News is a responsive web application that aggregates news from various sources through the News API. This document provides comprehensive information about the project's features, setup, and usage.

1. A professional structure with numbered sections and subsections
2. Detailed information about the project architecture, features, and implementation
3. Installation and configuration instructions with code examples
4. Usage guidelines with interface descriptions
5. API integration details
6. Team information and contributions
7. Future development plans
8. License and copyright information

I've used placeholder images where screenshots would normally appear. For a complete documentation, you would want to:

1. Replace placeholder images with actual screenshots of your application
2. Update any repository URLs with your actual GitHub repository
3. Verify that the API key instructions match your current implementation
4. Add any additional institution-specific information

1.2 Intended Audience

- Development team members
- Project evaluators
- Future contributors
- Academic faculty

2. Project Overview

2.1 Project Description

InsightStream News is a modern news aggregation website developed as part of the academic curriculum at DRBCCC Hindu College. The platform demonstrates practical implementation of web development concepts including API integration, responsive design, and dynamic content rendering.

2.2 Objectives

- Create a user-friendly news browsing experience
- Demonstrate API integration in a production environment

- Implement responsive design principles
- Showcase error handling and state management
- Apply modern JavaScript practices

2.3 Target Audience

- General news readers
- Students and faculty
- Web development enthusiasts

3. Features

3.1 Core Features

3.1.1 Multi-Category News Browsing

The application supports six primary news categories:

- General
- Business
- Technology
- Politics
- Entertainment
- Chennai (local news)

3.1.2 Search Functionality

Users can search for specific news topics through:

- Keyword input field
- Search button
- Enter key functionality

The search query automatically appends "news" to improve result relevance.

3.1.3 Dynamic Content Rendering

News articles are displayed in a responsive card layout with:

- Featured images
- Headlines
- Source information

- Publication timestamp
- Brief description

3.1.4 Error Handling

The application includes comprehensive error states:

- Loading indicators
- No-results messaging
- API error notifications
- Image fallback mechanism

3.2 Unique Capabilities

- **Mobile-First Design:** Fully responsive across all device sizes
- **Performance Optimization:** Efficient API calls and image handling
- **User Experience:** Intuitive interface with visual feedback
- **Local Focus:** Dedicated Chennai news category

4. Technical Architecture

4.1 Technology Stack

- **CSS3:** Custom styling with CSS variables
- **JavaScript:** ES6+ features
- **Font Awesome 6.0:** Icon library

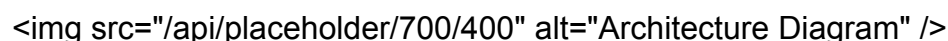
4.1.2 APIs & Services

- **News API:** Primary data source
- **Fetch API:** Native HTTP requests

4.1.3 Development Tools

- **Git:** Version control
- **VS Code:** Development environment

4.2 Architecture Overview

The diagram illustrates the client-side architecture of the application, showing the flow of data and components involved in the user experience.

The application follows a client-side architecture:

1. HTML Layer:

- Semantic document structure

- Navigation component
- News card template
- Search functionality
- Footer information

2. CSS Layer:

- Responsive grid system
- Flexbox layouts
- CSS variables for theming
- Media queries for responsiveness

3. JavaScript Layer:

- API integration
- DOM manipulation
- Event handling
- Error management

4. Data Flow:

- User interaction triggers category/search selection
- Application makes API request
- Response data is processed and validated
- News cards are dynamically generated
- Content is rendered to the DOM

5. Installation Guide

5.1 Prerequisites

- Modern web browser (Chrome 90+, Firefox 88+, Edge 90+)
- News API account (for API key)
- Basic knowledge of HTML/CSS/JS (for customization)

5.2 Setup Instructions

Clone the Repository

bash

```
https://github.com/SanthoshRaj9114/InsightStream-Navigate-the-News-Landscape.git
```

```
1. Cd InsightStream
```

2. API Key Configuration

- Register at [News API](#)
- Locate the API key variable in `script.js`:

```
javascript
```

Copy

```
const API_KEY = "your_api_key_here";
```

- Replace with your actual API key

3. Launch the Application

- Open `index.html` in a web browser

4. Verify Installation

- Technology category news should load by default
- Category navigation should be functional
- Search functionality should return results

5.3 Troubleshooting

Problem	Possible Fix
News not displayed	Check that the API key is valid and the quota has not been exceeded.
Unable to switch categories	Look for JavaScript errors in the browser console.
Images not loading	Check internet connection; fallback images should be displayed.

Search not working Ensure event listeners are correctly attached.

6. Configuration

6.1 API Configuration

The News API integration is configured in `script.js`:

javascript

```
const API_KEY = "a2263ce9b4a24ebfa0f59710526a65c6"; // Replace  
with your API key  
const url = "https://newsapi.org/v2/everything?q=";
```

6.2 Theme Configuration

CSS variables in `styles.css` control the application theme:

css

Copy

```
:root {
  --primary-text-color: #183b56;
  --secondary-text-color: #577592;
  --accent-color: #2294ed;
  --accent-color-dark: #1d69a3;
  --background-light: #f3faff;
  --shadow-color: #bbd0e2;
}
```

6.3 Category Configuration

To add or modify news categories:

1. Add a new list item in the navigation section of `index.html`:

html

Copy

```
<li class="hover-link nav-item" id="new-category"
onClick="onNavItemClick('new-category') ">New
Category</li>
```

2. Update the footer category section for consistency.

7. Usage Instructions

7.1 Basic Navigation

``

7.1.1 Category Selection

Click any category in the top navigation bar

Active category is highlighted with accent color

Default category is Technology

7.1.2 Search Functionality

Enter keywords in the search field

Press Enter or click Search button

Results display automatically

No category remains active during search results

7.1.3 Article Interaction

Click any news card to open original article

Hover over cards for visual feedback

7.2 Interface Elements

7.2.1 News Card

- Each card displays:
- Featured image
- Article title
- Source name
- Publication date/time
- Brief description

7.2.2 Status Indicators

- Loading message during API requests
- Error messages for failed requests
- No-results message for empty searches

8. API Integration

8.1 News API Overview

The application uses News API to fetch current news articles.

8.2 API Implementation

javascript

```
if (isLoading) return;

isLoading = true;
showLoadingIndicator();

const encodedQuery = encodeURIComponent(`${query}
news`);
const res = await
fetch(`${url}${encodedQuery}&apiKey=${API_KEY}`);
if (!res.ok) {
throw new Error(`HTTP error! status:
```

```

    ${res.status}`);
  }

  const data = await res.json();

  if (data.status === "error") {
    throw new Error(data.message);
  }

  bindData(data.articles);
} catch (error) {
  showError(error.message);
} finally {
  isLoading = false;
  hideLoadingIndicator();
}
}

```

8.3 Response Structure

```

json
{
  "articles": [
    {
      "source": {"id": null, "name": "News-source"},
      "author": "Author Name",
      "title": "News Title",
      "description": "News summary",
      "url": "Article URL",
      "urlToImage": "Image URL",
      "publishedAt": "ISO8601 Date",
      "content": "Full content"
    }
  ]
}

```

8.4 Rate Limiting

- Free tier: 100 requests/day
- Consider implementing caching for production use

HTML Code:

1. Document Type and Head Section

The document starts with the `<!DOCTYPE html>` declaration, indicating that this is an HTML5 document.

Head Section

```
html
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title> Insight Stream News</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.
0/css/all.min.css">
  <script src="script.js" defer></script>
</head>
```

The `<meta charset="UTF-8">` ensures proper character encoding.

`<meta name="viewport" content="width=device-width, initial-scale=1.0">` ensures the page is responsive.

`<title> Insight Stream News</title>` sets the page title.

`<link rel="stylesheet" href="styles.css">` links the external CSS file for styling.

`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">` imports Font Awesome icons.

`<script src="script.js" defer></script>` links an external JavaScript file and defers execution until the page loads.

2. Body Section

The `<body>` contains the visible content of the webpage.

2.1 Navigation Bar

html

```
<nav>
  <div class="main-nav">
    <div class="left-section">
      <a href="" class="company-logo">
          </a>
      <h1 class="site-title">InsightStream News</h1>  </div>
      <div class="nav-links">
        <ul>
          <li class="hover-link nav-item" id="general"
onClick="onNavItemClick('general')">General</li>
          <li class="hover-link nav-item" id="business"
onClick="onNavItemClick('business')">Business</li>  <li
class="hover-link nav-item" id="technology"
onClick="onNavItemClick('technology')">Technology</li>  <li
class="hover-link nav-item" id="politics"
onClick="onNavItemClick('politics')">Politics</li>  <li
class="hover-link nav-item" id="entertainment"
onClick="onNavItemClick('entertainment')">Entertainment</li>
          <li class="hover-link nav-item" id="chennai"
onClick="onNavItemClick('chennai')">Chennai</li>  </ul>
        </div>
        <div class="search-bar">
          <input type="text" class="news-input"
id="search-text" placeholder="e.g. Science">
          <button class="search-button"
id="search-button">Search</button>
        </div>
      </div>
    </nav>
```

- The **navigation bar** contains:
 - A **logo** and site title.
 - Navigation **menu links** (General, Business, Technology, etc.), each having an `onClick` event calling

onNavItemClick(category).

- A **search bar** with an input field (#search-text) and a button (#search-button) for user searches.

2.2 Main Content Section

```
html
CopyEdit
<main>
  <div class="cards-container container flex"
id="cardscontainer">
    </div>
</main>
```

- The cards-container is where **news articles** will be dynamically loaded.

2.3 News Card Template

```
html

<template id="template-news-card">
  <div class="card">
    <div class="card-header">
      <img
src=""
alt="news image"
id="news-img"

onerror="this.src='./images/fallback-image.jpg'"
>
    </div>
    <div class="card-content">
      <h3 id="news-title">InsightStream</h3>  <h6
class="news-source"
id="news-source">InsightStream news</h6>
      <p class="news-desc" id="news-desc">Lorem ipsum dolor sit
amet consectetur adipisicing elit.</p>  </div>
    </div>
```

</template>

- This **template** is used for dynamically adding news articles.
- The **image** has an `onerror` event, which replaces a broken image with a fallback image (`fallback-image.jpg`).
- It contains:
 - `news-title`: The article title.
 - `news-source`: The source of the article.
 - `news-desc`: The article description.

2.4 Footer Section

html

```
<footer>
```

```
  <div class="footer-main">
```

```
    <div class="footer-container">
```

```
      <div class="footer-section">
```

```
        <h3>About InsightStream News</h3>  <p>Your trusted source for  
the latest news and updates from around the world.</p>
```

```
      </div>
```

```
      <div class="footer-section">
```

```
        <h3>Academic Credits</h3>
```

```
        <p>DRBCCC Hindu College</p>
```

```
        <p>Project Guide: Asst. Prof. Jansi</p>  <p>News API  
Integration Project</p>  </div>
```

```
      <div class="footer-section">
```

```
        <h3>Project Team</h3>
```

```
        <ul>
```

```
          <li>Prashanth N (Lead)</li>  <li>Santhosh Raj  
V</li>
```

```
          <li>Magesh Kumar T</li>
```

```
          <li>Saran P</li>
```

```
          <li>Kamalakannan M M</li>
```

```
        </ul>
```

```
      </div>
```

```
      <div class="footer-section">
```

```
        <h3>Categories</h3>
```

```
        <ul>
```

```
          <li><a href="#"
```

```

onclick="onNavItemClick('technology')">Technology</a></li>
<li><a href="#"
onclick="onNavItemClick('business')">Business</a></li>
<li><a href="#"
onclick="onNavItemClick('entertainment')">Entertainment</a></li>
</li>
<li><a href="#"
onclick="onNavItemClick('politics')">Politics</a></li>
<li><a href="#"
onclick="onNavItemClick('chennai')">Chennai</a></li>
</ul>
</div>
<div class="footer-section">
<h3>Connect With Us</h3>
<div class="social-links">
<a href="#"><i class="fab
fa-facebook"></i></a>
<a href="#"><i class="fab
fa-twitter"></i></a>
<a href="#"><i class="fab
fa-instagram"></i></a>
</div>
</div>
</div>
</div>
<div class="footer-bottom">
<p>&copy; 2025 InsightStream News. All rights
reserved.</p>
</div>
</footer>

```

The **footer** contains:

- **About section** explaining InsightStream News.
- **Academic credits**, mentioning DRBCCC Hindu College and the Project Guide: Asst. Prof. Jansi.
- **Project team** listing the contributors.
- **Categories** with quick navigation links.

Java script

Code Breakdown

1. API Key and Base URL

```
javascript
const API_KEY = "a2263ce9b4a24ebfa0f59710526a65c6";
const url = "https://newsapi.org/v2/everything?q=";
```

API_KEY: The key used to authenticate requests with **NewsAPI**.

url: The base URL to fetch news data.

2. State Variables

```
javascript
let isLoading = false;
let curSelectedNav = null;
```

isLoading: Prevents multiple API calls when a request is already in progress. **curSelectedNav**: Stores the currently active navigation item.

3. Initial News Fetch on Page Load

```
javascript
window.addEventListener("load", () => {
  fetchNews("Technology");
  const techNav = document.getElementById("technology");
  techNav?.classList.add("active");
  curSelectedNav = techNav;});
```

Event Listener: Triggers when the page loads.

fetchNews("Technology"): Fetches news related to "Technology" by default.

Activates the Technology Tab: Highlights the "Technology" navigation item.

4. Fetching News from the API

```
javascript
async function fetchNews(query) {
  try {
    if (isLoading) return;

    isLoading = true;
    showLoadingIndicator();

    const encodedQuery = encodeURIComponent(`${query}
news`);
    const res = await
fetch(`${url}${encodedQuery}&apiKey=${API_KEY}`);
    if (!res.ok) {
      throw new Error(`HTTP error! status:
${res.status}`);
    }

    const data = await res.json();

    if (data.status === "error") {
      throw new Error(data.message);
    }

    bindData(data.articles);
  } catch (error) {
    showError(error.message);
  } finally {
    isLoading = false;
    hideLoadingIndicator();
  }
}
```

Prevents multiple fetches: If a request is already in progress (isLoading is true), it exits.

Encodes search query: Prevents special characters from breaking the API request.

Fetches news from API: Using `fetch()`, it sends a request and awaits the response.

Handles errors: If the response is invalid or API returns an error, it shows an error message.

Calls `bindData()`: Updates the UI with fetched news articles.

Resets `isLoading`: Ensures the page can make another request after the current one completes.

5. UI Feedback Functions

Loading Indicator

javascript

```
function showLoadingIndicator() {
  const cardsContainer = document.getElementById("cards
container");
  cardsContainer.innerHTML = '<div class="loading">Loading
news...</div>';
}
```

```
function hideLoadingIndicator() {
  const loadingDiv = document.querySelector(".loading");
  if (loadingDiv) {
    loadingDiv.remove();
  }
}
```

Displays "Loading news..." while fetching data.

Removes the loading message when data is loaded.

Error Handling

javascript

```
function showError(message) {
  const cardsContainer = document.getElementById("card
```

```

container");
  cardsContainer.innerHTML = `

Displays an error message if fetching news fails.



## 6. Populating the UI with News Articles



```

javascript
function bindData(articles) {
 const cardsContainer = document.getElementById("cards
container");
 const newsCardTemplate =
document.getElementById("template-news-card");

 cardsContainer.innerHTML = "";

 if (!articles || articles.length === 0) {
 cardsContainer.innerHTML = '<div class="no-results">No news
found</div>';
 return;
 }

 articles.forEach((article) => {
 if (!article.urlToImage || !article.description)
return;
 const cardClone =
newsCardTemplate.content.cloneNode(true);
 fillDataInCard(cardClone, article);
 cardsContainer.appendChild(cardClone);
 });
}

```



Clears old news data before displaying new articles.


```

Handles no results: If no articles are found, it displays "No news found".
Loops through articles and creates a card for each one.

Skips articles that lack an image or description.

7. Populating Individual News Cards

```
javascript
function fillDataInCard(cardClone, article)
{
    const newsImg = cardClone.querySelector("#news-img"); const
newsTitle = cardClone.querySelector("#news-title"); const
newsSource =
cardClone.querySelector("#news-source");
    const newsDesc = cardClone.querySelector("#news-desc");
    newsImg.src = article.urlToImage;
    newsImg.onerror = function() {
    this.src = './images/fallback-image.jpg'; };

    newsTitle.innerHTML = truncateText(article.title, 60);
newsDesc.innerHTML = truncateText(article.description,
150);

    const date = new
Date(article.publishedAt).toLocaleString("en-IN", {
timezone: "Asia/Kolkata",
    day: 'numeric',
    month: 'short',
    year: 'numeric',
    hour: '2-digit',
    minute: '2-digit'
});

    newsSource.innerHTML = `${article.source.name} · ${date}`;
cardClone.firstElementChild.addEventListener("click", () =>
{
```

```
window.open(article.url, "_blank");
});
}
```

8. Truncate Long Text

```
javascript
function truncateText(text, maxLength) {
  if (!text) return '';
  return text.length > maxLength ? `${text.slice(0,
maxLength)}...` : text;
}
```

9. Handling Navigation Clicks

```
javascript
function onNavItemClick(id) {
  const navItem = document.getElementById(id);
  if (navItem === curSelectedNav) return;

  fetchNews(id);
  curSelectedNav?.classList.remove("active");
  curSelectedNav = navItem;
  curSelectedNav.classList.add("active");
}
```

Fetches news when a navigation item is clicked.

10. Search Functionality

```
javascript
const searchButton = document.getElementById("search-button");
const searchText = document.getElementById("search-text");

searchText.addEventListener("keypress", (e) => {
  if (e.key === "Enter") {
```

```

    const query = searchText.value.trim();
    if (query) {
      fetchNews(query);
      curSelectedNav?.classList.remove("active");
    }
    curSelectedNav = null;
  });

  searchButton.addEventListener("click", () => {
    const query = searchText.value.trim();
    if (query) {
      fetchNews(query);
      curSelectedNav?.classList.remove("active");
    }
    curSelectedNav = null;
  });

```

Styles.css

Global Styles

```

* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

```

Resets default browser styles by removing padding and margin from all elements.

Ensures consistent box-sizing (border-box) so padding and borders don't affect element dimensions.

CSS Variables (Theme Colors)

```

:root {
  --primary-text-color: #183b56;
  --secondary-text-color: #577592;
  --accent-color: #2294ed;
  --accent-color-dark: #1d69a3;
}

```

```
--background-light: #f3faff;
--shadow-color: #bbd0e2;
}
```

Defines **CSS variables** for colors, making the theme easily customizable.

Base Styles

```
body {
  font-family: "Poppins", sans-serif;
  color: var(--primary-text-color);
  background-color: #f8f9fa;
}
```

Sets **default font** (Poppins).

Uses **theme colors** for text and background.

```
p {
  color: var(--secondary-text-color);
  line-height: 1.4rem;
}
```

Styles `<p>` tags with **secondary text color** and **line spacing**.

```
a {
  text-decoration: none;
}
```

Removes **underline** from links.

```
ul {
  list-style: none;
}
```

Removes **bullets** from unordered lists.

utility Classes

```
.flex {
  display: flex;
  align-items: center;
}
```

```
}
```

A **helper class** to easily apply `display: flex` to

elements. `.container` {

```
max-width: 1280px;
```

```
margin-inline: auto;
```

```
padding: 0 20px;
```

```
}
```

Centers the content and ensures it doesn't exceed 1280px width.

Navigation Bar (nav)

```
nav {
```

```
background-color: var(--background-light);
```

```
box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
```

```
position: fixed;
```

```
top: 0;
```

```
z-index: 99;
```

```
left: 0;
```

```
right: 0;
```

```
height: auto;
```

```
}
```

Makes the navigation bar **fixed at the top** (`position:`

`fixed`). Adds a **shadow effect** for a **floating look**.

```
.main-nav {
```

```
display: flex;
```

```
align-items: center;
```

```
justify-content: space-between;
```

```
padding: 15px 40px;
```

```
max-width: 1400px;
```

```
margin: 0 auto;
```

```
}
```

Aligns **navigation items horizontally** using `display: flex`.

Provides **spacing** using padding.

Logo and Title

```
.company-logo img {
```



```
width: 100px;
height: auto;
}
```

Ensures the **logo** maintains proportions.

```
.site-title {
font-size: 1.5rem;
font-weight: 600;
margin-left: 10px;
}
```

Styles **website name** with **larger, bold text**.

Navigation Links

```
.nav-links ul {
display: flex;
gap: 32px;
align-items: center;
}
```

Displays **menu items** in a **horizontal row** with **equal spacing**.

```
.hover-link {
cursor: pointer;
padding: 8px 16px;
border-radius: 4px;
transition: all 0.3s ease;
font-size: 15px;
}
```

Adds a **hover effect** to navigation links.

```
.hover-link:hover {
color: var(--accent-color);
background-color: rgba(34, 148, 237, 0.1);
}
```

Changes text color and **adds a background highlight** on hover.

```
.nav-item.active {  
  color: var(--accent-color);  
  font-weight: 500;  
}
```

Highlights the **active page**.

Search Bar

```
.search-bar {  
  margin-left: auto;  
  display: flex;  
  align-items: center;  
  gap: 12px;  
  background-color: white;  
  border-radius: 4px;  
  padding: 6px;  
}
```

Positions the search bar to the right. Adds a **white background** with rounded edges.

```
.news-input {  
  width: 200px;  
  padding: 8px 16px;  
  border: 1px solid #e0e0e0;  
  border-radius: 4px;  
}
```

Styles **input field** for user queries.

```
.search-button {  
  background-color: var(--accent-color);  
  color: white;  
  padding: 8px 24px;  
  border-radius: 4px;  
  cursor: pointer;  
  font-weight: 500;  
  transition: all 0.3s ease;  
}
```

Styles **search button** with a blue background.

```
.search-button:hover {  
  background-color: var(--accent-color-dark);  
  transform: translateY(-1px);  
}
```

Hover effect: Darkens the button and **lifts it slightly**.

Main Content

```
main {  
  padding-block: 20px;  
  margin-top: 100px;  
}
```

Adds **padding** and ensures content **does not overlap the navbar**.

Cards (News Articles)

```
.cards-container {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
  gap: 24px;  
  padding: 20px;  
  max-width: 1280px;  
  margin: 0 auto;  
}
```

Uses **CSS Grid** to display news articles **in a responsive layout**.

```
.card {  
  background-color: white;  
  border-radius: 8px;  
  overflow: hidden;  
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
  transition: all 0.3s ease;  
}
```

Styles individual **news cards**.

```
.card-header img {  
  width: 100%;  
  height: 200px;  
  object-fit: cover;  
}
```

Ensures **news images** fit properly.

Footer Styles

```
footer {  
  background-color: #1a1a1a;  
  color: #ffffff;  
  margin-top: 50px;  
}
```

Dark-themed footer for contrast.

```
.footer-container {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 40px;  
}
```

Uses **CSS Grid** for a **structured footer layout**.

```
.social-links a {  
  color: #ffffff;  
  font-size: 20px;  
  transition: color 0.3s ease;  
}
```

Styles **social media icons**.

Responsive Footer (Mobile View)

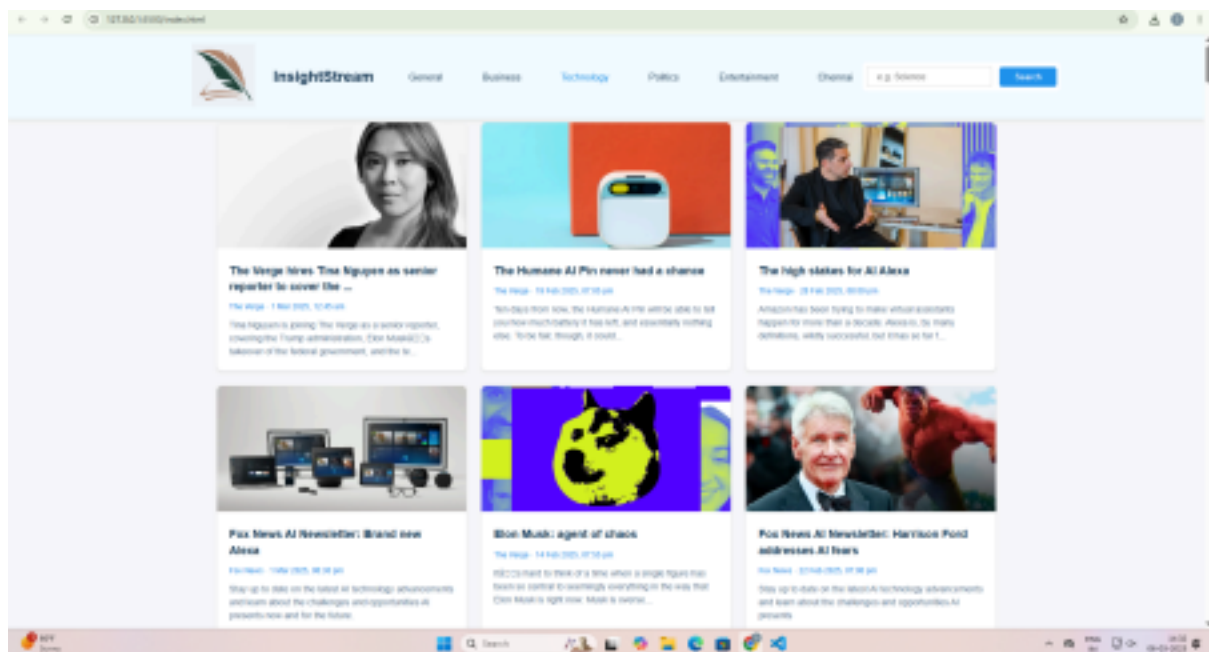
```
@media (max-width: 768px) {  
  .footer-container {  
    grid-template-columns: 1fr;  
    text-align: center;  
  }  
}
```

```
.social-links {  
  justify-content: center;  
}  
}
```

Adjusts footer layout for smaller screens by **stacking sections vertically**.

10. Screen layout

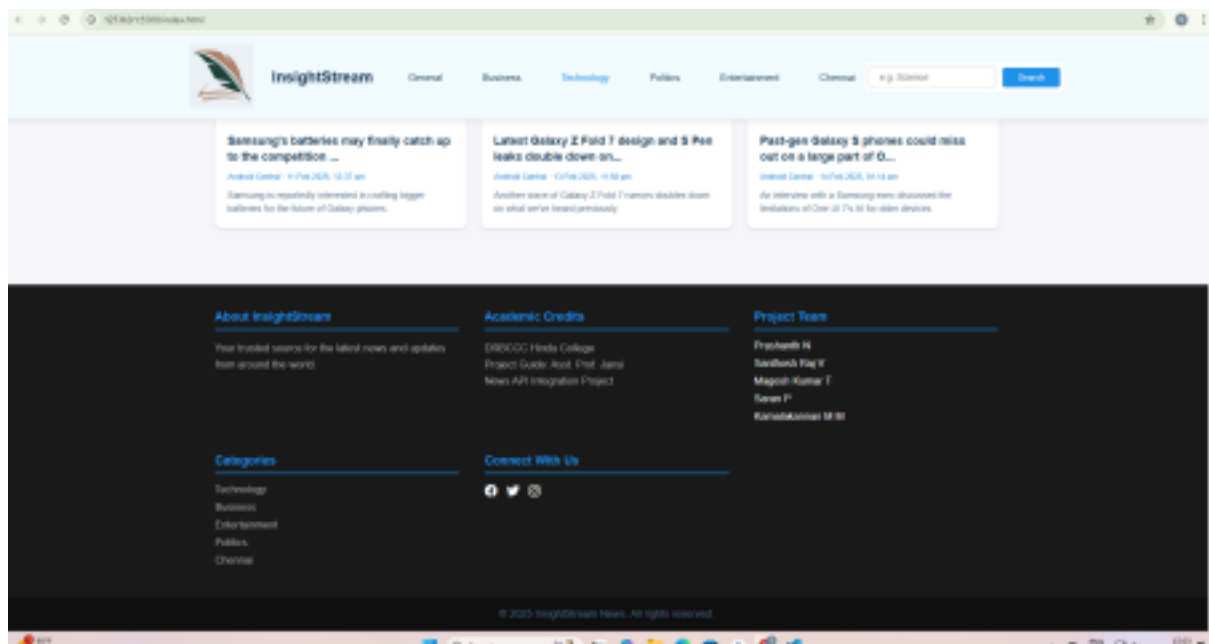
Home page



News redirect page



Footer Page



Demo video:

https://drive.google.com/file/d/1FWQQQfObyTEj_DzkSzghErzzyWGH-s6E/view?pli=1

11. Future Development

11.1.1 Planned Enhancements

11.1.1 User Features

- User accounts and authentication
- Personalized news preferences
- Saved articles functionality
- Reading history

11.1.2 Content Improvements

- Advanced filtering options
- Source selection
- Date range filtering
- Content recommendations

11.1.3 Technical Enhancements

- Progressive Web App implementation
- Offline capability
- Push notifications
- Performance optimizations

11.2 Known Limitations

- API request quota (100/day) on free tier
- No persistence between sessions
- Limited to News API's available sources
- No user customization options

12. Conclusion

InsightStream News is a dynamic and responsive news aggregation platform designed to provide users with a seamless news browsing experience across multiple categories. With the integration of the News API and a mobile-first approach, the application demonstrates practical use of modern web development techniques, including API integration, responsive design, and real-time content updates.

The platform offers a user-friendly interface, complete with features like multi-category browsing, search functionality, and dynamic content rendering. Its efficient performance and error-handling mechanisms ensure a smooth user experience, even under varying conditions.

Through this project, the development team has successfully implemented key industry practices while providing a valuable resource for news enthusiasts, students, and faculty. Future developments and features will continue to enhance the user experience, making InsightStream News a reliable and attractive platform for staying updated with the latest news.

The project not only serves as a robust academic example of web development but also as a stepping stone for future contributors to build upon and expand its capabilities.