Project Report: REST API

Project Title: Global News Hub

Name: BIDYACHANDRA MAIBRAM

Register No.: 23BTCE203

Date: September 9, 2025

Project Report on Global News Hub

1. Introduction

For my project, I built the Global News Hub, a React-based web application that provides users with real-time news updates from around the world. I used the GNews API to fetch news articles and organized them into categories such as general, technology, sports, business, entertainment, and health. I designed the platform with an easy-to-use interface so that users can browse headlines by category or search for specific topics of interest.

2. My Objectives

My main objectives for this project were:

- To build a modern, responsive, and user-friendly news portal.
- To fetch real-time global news by integrating the GNews API.
- To implement a feature allowing users to filter news by categories.
- To provide a search functionality for finding topic-specific news.

- To ensure the application handles missing data gracefully by using placeholders.

3. Tools and Technologies I Used

- **Frontend Framework:** React.jsx
- **Styling:** Tailwind CSS
- **API Provider:** GNews API (https://gnews.io/)
- **Programming Language:** JavaScript (ES6)
- **Package Manager:** npm (assumed)
- **Build Tool:** Vite (assumed)

4. My System Design

4.1 Functional Components I Created

I structured the application using several key functional components:

- **ArticleCard Component:** This component I created displays individual news articles with a title, description, image, source, date, and a "Read More" link. I made sure it uses a placeholder image if an article has no image to maintain UI consistency.
- **LoadingSpinner Component:** I built this to show a spinner animation whenever news is being fetched from the API, providing clear visual feedback.
- **StatusMessage Component:** I use this component to display any error messages, empty results, or other important notifications. I styled it differently based on the message type (e.g., error vs. info).
- **App Component (Main):** This is the core of my application. It manages the application state (articles, loading status, errors), handles the API calls, provides the category filters and search functionality, and displays the articles in a responsive grid.

5. Features I Implemented

- **Category Filtering** – I enabled users to switch between categories (general, technology, sports, business, entertainment, health).

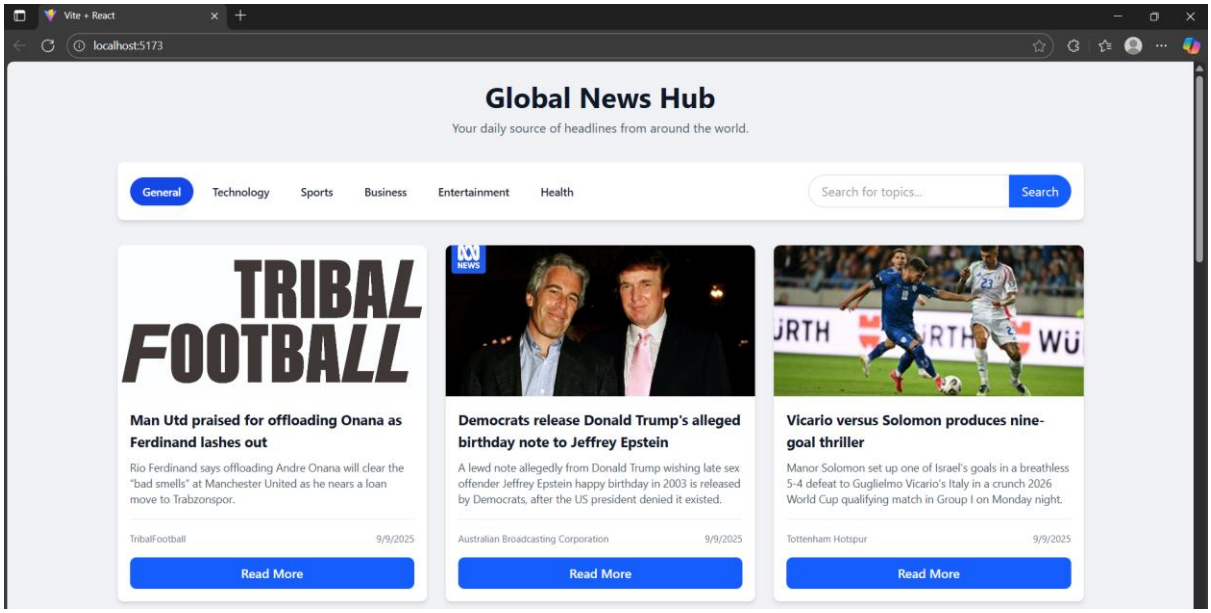
- **Search Functionality** – I added a search bar that allows users to look for news articles by keywords.
- **Error Handling** – The app displays appropriate error messages if the API key is missing or invalid.
- **Loading Animation** – I included a loading spinner to provide feedback when the app is fetching news.
- **Responsive UI** – I designed the layout to adapt seamlessly to both desktop and mobile screens.

6. My Project's Workflow

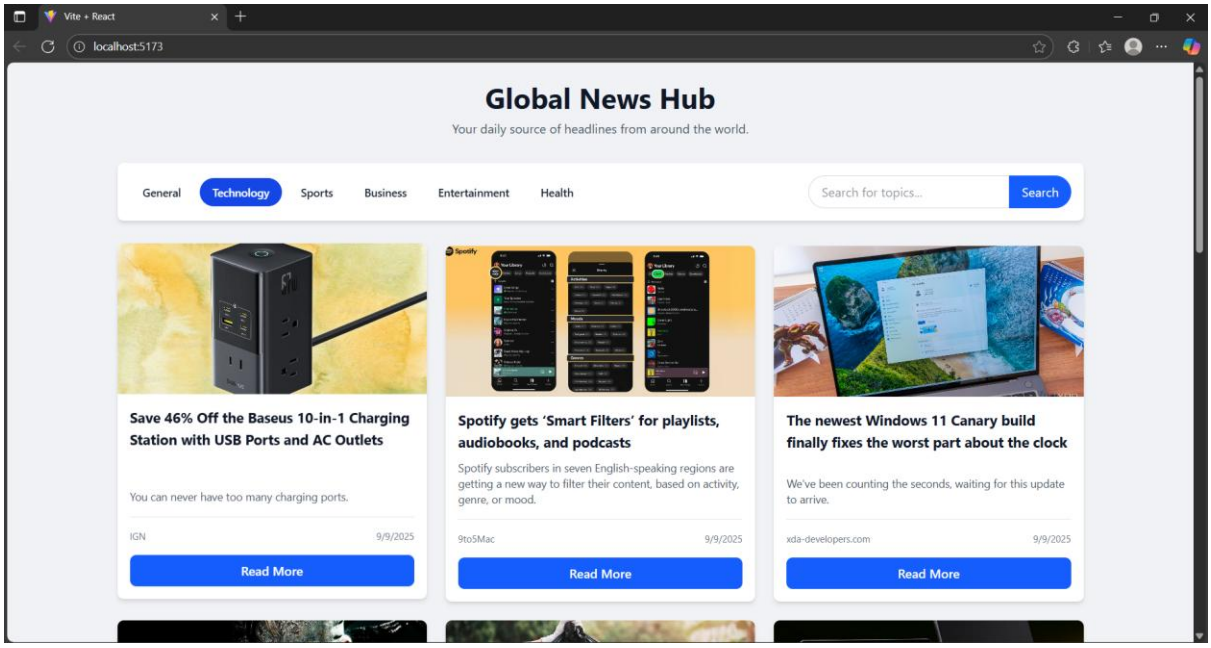
On initial load, I have the app fetch the top headlines in the "General" category. Users can click on different category buttons, and the app then dynamically fetches the relevant headlines for that topic. When a user enters a search term and submits it, the app fetches results based on that query. If the API key is invalid, the network fails, or no results are found, I made sure an appropriate status message is displayed. To let users read the full article, I added a “Read More” button that opens the original news source in a new tab.

7. Screenshots

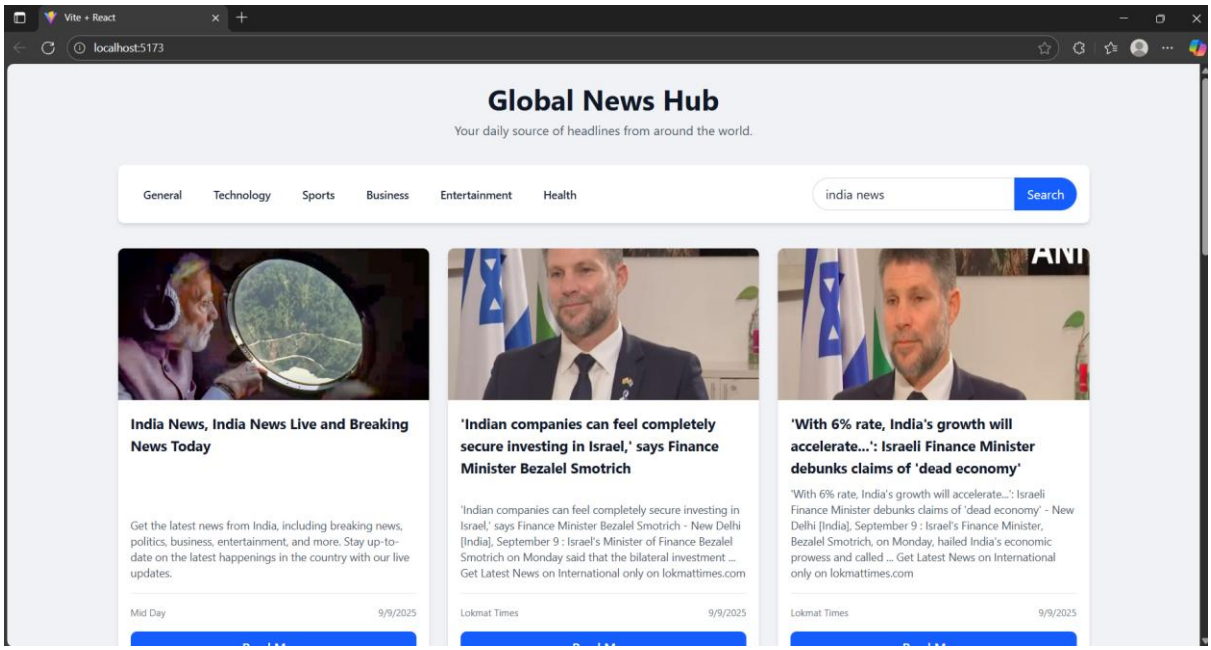
Screenshot 1: *Homepage with general headlines.*



Screenshot 2: *Category selection showing "Technology" news.*



Screenshot 3: *Search results for a keyword.*



8. Code Snippet

```
import React, { useState, useEffect, useCallback } from 'react';

// --- IMPORTANT ---
// Get your free API key from https://gnews.io/
const API_KEY = '91c7c490f6071f00202ee4c65e39e210';

const API_BASE_URL = 'https://gnews.io/api/v4';
const CATEGORIES = ['general', 'technology', 'sports', 'business', 'entertainment', 'health'];

// Reusable component for a single news article
const ArticleCard = ({ article }) => {
  // Use a placeholder image if the article image is missing or fails to load
  const imageUrl = article.image ||
`https://placehold.co/600x400/gray/white?text=${encodeURIComponent(article.source.nam
e)}`;

  const handleImageError = (e) => {
    e.target.onerror = null; // prevent infinite loop
    e.target.src = `https://placehold.co/600x400/gray/white?text=Image+Not+Found`;
  };

  return (
    <div className="bg-white rounded-lg shadow-md overflow-hidden flex flex-col
transform hover:-translate-y-1 transition-transform duration-300">
      <img src={imageUrl} alt={article.title} className="h-48 w-full object-cover"
onError={handleImageError} />
      <div className="p-4 flex flex-col flex-grow">
        <h3 className="text-lg font-bold text-gray-900 mb-2 flex-grow">{article.title}</h3>
        <p className="text-sm text-gray-600 mb-4">{article.description || 'No description
available.'}</p>
        <div className="mt-auto pt-4 border-t border-gray-200">
          <div className="flex justify-between items-center text-xs text-gray-500">
            <span>{article.source.name}</span>
            <span>{new Date(article.publishedAt).toLocaleDateString()}</span>
          </div>
          <a href={article.url} target="_blank" rel="noopener noreferrer"
className="block w-full text-center bg-blue-600 text-white font-semibold py-2 mt-4
rounded-lg hover:bg-blue-700 transition">
            Read More
          </a>
        </div>
      </div>
    </div>
  );
```

```
};

// Loading spinner component
const LoadingSpinner = () => (
    <div className="text-center py-10">
        <div className="spinner mx-auto" style={{
            border: '4px solid rgba(0, 0, 0, 0.1)',
            width: '40px',
            height: '40px',
            borderRadius: '50%',
            borderLeftColor: '#1d4ed8',
            animation: 'spin 1s ease infinite'
        }}></div>
        <p className="mt-4 text-gray-600">Fetching the latest headlines...</p>
        <style>{`
            @keyframes spin {
                to { transform: rotate(360deg); }
            }
        `}</style>
    </div>
);

// Component for displaying status messages (error, no results)
const StatusMessage = ({ message, type }) => {
    const baseClasses = 'text-center py-10 rounded-lg';
    const typeClasses = type === 'error'
        ? 'bg-red-100 text-red-700 p-4'
        : 'text-gray-600';
    return (
        <div className={`${baseClasses} ${typeClasses}`} dangerouslySetInnerHTML={{ __html:
message }}></div>
    );
};

// Main App Component
function App() {
    const [articles, setArticles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    const [activeCategory, setActiveCategory] = useState('general');
    const [searchTerm, setSearchTerm] = useState('');

    const fetchNews = useCallback(async (category, query) => {
        if (API_KEY === 'YOUR_API_KEY_HERE') {
            setError('<strong>Action Required:</strong> Please add your GNews API key.');
            setLoading(false);
            return;
```

```
        }

        setLoading(true);
        setError(null);
        setArticles([]);

        let url;
        if (query) {
            url =
`${API_BASE_URL}/search?q=${encodeURIComponent(query)}&lang=en&apikey=${API_KEY}
`;
        } else {
            url = `${API_BASE_URL}/top-
headlines?topic=${category}&lang=en&apikey=${API_KEY}`;
        }

        try {
            const response = await fetch(url);
            const data = await response.json();

            if (!response.ok) {
                throw new Error(data.errors?.[0] || `HTTP error! Status: ${response.status}`);
            }

            setArticles(data.articles || []);
        } catch (err) {
            console.error("Failed to fetch news:", err);
            setError(`<strong>Error:</strong> ${err.message}. Please check your API key and
network connection.`);
        } finally {
            setLoading(false);
        }
    }, []);

    // Effect to fetch news when category or search term changes
    useEffect(() => {
        if (searchTerm) {
            fetchNews(null, searchTerm);
        } else {
            fetchNews(activeCategory, null);
        }
    }, [activeCategory, searchTerm, fetchNews]);

    const handleCategoryClick = (category) => {
        setSearchTerm('');
        setActiveCategory(category);
    };
```

```
  const handleSearchSubmit = (e) => {
    e.preventDefault();
    const newSearchTerm = e.target.elements['search-input'].value.trim();
    if (newSearchTerm) {
      setActiveCategory(null);
      setSearchTerm(newSearchTerm);
    }
  };

  return (
    <div className="bg-[#f0f2f5] min-h-screen font-sans text-gray-800">
      <div className="container mx-auto p-4 md:p-6">
        {/* Header */}
        <header className="text-center mb-6 md:mb-8">
          <h1 className="text-3xl md:text-4xl font-bold text-gray-900">Global News
Hub</h1>
          <p className="text-md text-gray-600 mt-2">Your daily source of headlines from
around the world.</p>
        </header>

        {/* Controls: Filters and Search */}
        <div className="bg-white p-4 rounded-lg shadow-md mb-6 md:mb-8 sticky top-2
z-10">
          <div className="flex flex-col md:flex-row md:items-center md:justify-between
gap-4">
            <div className="flex flex-wrap gap-2">
              {CATEGORIES.map(category => (
                <button
                  key={category}
                  onClick={() => handleCategoryClick(category)}
                  className={`px-4 py-2 text-sm font-medium rounded-full transition
hover:bg-gray-200 capitalize ${activeCategory === category ? 'bg-blue-700 text-white font-
semibold' : ''}`}
                >
                  {category}
                </button>
              ))}
            </div>
            <form onSubmit={handleSearchSubmit} className="flex w-full md:w-auto">
              <input
                type="text"
                id="search-input"
                name="search-input"
                placeholder="Search for topics..."
                className="w-full md:w-64 px-4 py-2 border border-gray-300 rounded-l-
full focus:outline-none focus:ring-2 focus:ring-blue-500"
```

```
                    />
                    <button type="submit" className="bg-blue-600 text-white px-4 py-2
rounded-r-full hover:bg-blue-700 transition">
                      Search
                    </button>
                  </form>
                </div>
              </div>

              {/* Main Content Area */}
              <main>
                {loading && <LoadingSpinner />}
                {error && <StatusMessage message={error} type="error" />}
                {!loading && !error && articles.length === 0 && (
                  <StatusMessage message="<strong>No articles found.</strong> Try a different
category or search term." type="info" />
                )}
                {!loading && !error && articles.length > 0 && (
                  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
                    {articles.map((article) => (
                      <ArticleCard key={article.url} article={article} />
                    ))}
                  </div>
                )}
              </main>
            </div>
          </div>
        );
}

export default App;
```

9. Conclusion

My Global News Hub project successfully demonstrates how I can integrate real-time APIs into a modern web application using React. I believe it provides a clean, responsive, and user-friendly way for people to access the latest news headlines worldwide. For the future, I think that with improvements such as user authentication, bookmarking favorite articles, or multilingual support, the application could evolve into a full-fledged, personalized news platform.