



COLLEGE CODE : 9604

COLLEGE NAME : CSI INSITUTE OF TECHNOLOGY

DEPARTMENT : INFORMATION TECHNOLOGY

STUDENT NM- ID : E2CFD434372F7C9876040500836D2C67

ROLL NO :960423205012

DATE : 7/10/2025

SUBMITTED BY,

NAME : JITHESH S

MOBILE NO: 9995035210



Phase 4 - Enhancements & Deployment : News Feed Application

1. Additional Features :

1.1 Product Search & Filter Implementation

Definition: This feature improves user experience by enabling users to dynamically filter the news feed by entering keywords, enhancing the discoverability of relevant news articles without requiring page reloads.

Implementation Details: The search functionality is implemented in JavaScript by attaching an event listener to the search input field, which filters the global state.articles array based on the entered keywords before re-rendering the news feed display.

Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-
```

```
scale=1" />
<title>Dynamic News Feed</title>
<style>
  body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f9fafb;
    color: #333;
    margin: 0;
    padding: 20px;
  }

  .search-bar {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-bottom: 25px;
    background: #fff;
    padding: 15px 20px;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgb(0 0 0 / 0.1);
  }

  .search-bar input[type="text"],
  .search-bar select,
  .search-bar input[type="date"] {
    flex: 1 1 150px;
    min-width: 150px;
    padding: 10px 12px;
    font-size: 1rem;
    border-radius: 6px;
    border: 1px solid #ccc;
    transition: border-color 0.3s ease;
```

```
}
```

```
.search-bar input[type="text"]:focus,  
.search-bar select:focus,  
.search-bar input[type="date"]:focus {  
  outline: none;  
  border-color: #0078d4;  
  box-shadow: 0 0 6px rgba(0, 120, 212, 0.3);  
}
```

```
.btn.detail-btn {  
  background-color: #0078d4;  
  color: white;  
  border: none;  
  border-radius: 6px;  
  padding: 10px 18px;  
  cursor: pointer;  
  transition: background-color 0.3s ease;  
  flex: 0 0 auto;  
}
```

```
.btn.detail-btn:hover {  
  background-color: #005a9e;  
}
```

```
h1.page-title {  
  font-weight: 700;  
  color: #0078d4;  
  margin-bottom: 20px;  
}
```

```
section.news-feed {
```

```
display: grid;
grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
gap: 20px;
}
```

```
article.news-item {
  background: white;
  border-radius: 10px;
  box-shadow: 0 2px 12px rgb(0 0 0 / 0.1);
  padding: 20px;
  transition: transform 0.2s ease, box-shadow 0.2s ease;
  display: flex;
  flex-direction: column;
}
```

```
article.news-item:hover {
  transform: translateY(-6px);
  box-shadow: 0 10px 20px rgb(0 0 0 / 0.15);
}
```

```
article.news-item h2 {
  margin-top: 0;
  font-size: 1.3rem;
  color: #222;
}
```

```
article.news-item p {
  color: #555;
  line-height: 1.4;
  margin: 6px 0;
}
```

```
article.news-item .meta {  
  font-size: 0.9rem;  
  color: #0078d4;  
  margin-bottom: 10px;  
}
```

```
p.no-articles-message {  
  font-style: italic;  
  color: #999;  
  grid-column: 1/-1;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="search-bar animated fadeInDown">
```

```
  <input type="text" id="news-search-input" placeholder="Search  
news (e.g., Politics, Sports)...">
```

```
  <select id="category-filter">
```

```
    <option value="">All Categories</option>
```

```
    <option value="politics">Politics</option>
```

```
    <option value="sports">Sports</option>
```

```
    <option value="technology">Technology</option>
```

```
    <option value="health">Health</option>
```

```
  </select>
```

```
  <input type="date" id="date-filter" placeholder="Filter by date">
```

```
  <input type="text" id="author-filter" placeholder="Filter by  
author">
```

```
<button id="clear-filters-btn" class="btn detail-btn">Clear</button>
</div>
```

```
<div id="appContainer"></div>
```

```
<script>
```

```
// Example global state with some sample articles
```

```
const state = {
```

```
  articles: [
```

```
    {
```

```
      title: "Latest Advances in Technology",
```

```
      content: "Discover the newest breakthroughs shaping the future  
of tech...",
```

```
      summary: "An intriguing overview of recent tech breakthroughs.",
```

```
      category: "technology",
```

```
      date: "2025-09-15",
```

```
      author: "Alice Johnson"
```

```
    },
```

```
    {
```

```
      title: "Politics Today: Election Updates",
```

```
      content: "Get the latest news on the upcoming elections and  
political shifts.",
```

```
      summary: "Current political updates and election coverage.",
```

```
      category: "politics",
```

```
      date: "2025-10-05",
```

```
      author: "Bob Smith"
```

```
    },
```

```
    {
```

```
      title: "Sports Highlights of the Week",
```

```
      content: "A recap of thrilling games and outstanding athletes'  
performances.",
```

```
      summary: "Weekly sports highlights and scores.",
```

```
    category: "sports",
    date: "2025-10-03",
    author: "Cara Lee"
  },
  {
    title: "Health Tips for a Better Life",
    content: "Explore some effective tips to improve your health and
wellness.",
    summary: "Practical advice for maintaining good health.",
    category: "health",
    date: "2025-09-30",
    author: "David Kim"
  }
];
```

```
function filterArticles(searchTerm, category, date, author) {
  const term = searchTerm.toLowerCase().trim();
  const categoryFilter = category.toLowerCase();
  const authorFilter = author.toLowerCase();

  return state.articles.filter(article => {
    const matchesTerm = article.title.toLowerCase().includes(term) ||
      article.content.toLowerCase().includes(term);
    const matchesCategory = categoryFilter ?
article.category.toLowerCase() === categoryFilter : true;
    const matchesAuthor = authorFilter ?
article.author.toLowerCase().includes(authorFilter) : true;
    const matchesDate = date ? (new Date(article.date).toDateString()
=== new Date(date).toDateString()) : true;
    return matchesTerm && matchesCategory && matchesAuthor &&
matchesDate;
  });
}
```



```
});  
}
```

```
function renderArticles() {  
  const searchInput = document.getElementById('news-search-  
input');  
  const categorySelect = document.getElementById('category-filter');  
  const dateInput = document.getElementById('date-filter');  
  const authorInput = document.getElementById('author-filter');  
  
  const searchTerm = searchInput ? searchInput.value : "";  
  const selectedCategory = categorySelect ? categorySelect.value : "";  
  const selectedDate = dateInput ? dateInput.value : "";  
  const selectedAuthor = authorInput ? authorInput.value : "";  
  
  const filteredArticles = filterArticles(searchTerm, selectedCategory,  
selectedDate, selectedAuthor);  
  
  const appContainer = document.getElementById('appContainer');  
  appContainer.innerHTML = `  
    <h1 class="page-title animated fadeInRight">All News</h1>  
    <section class="news-feed">  
      ${filteredArticles.length > 0 ? filteredArticles.map((article, index)  
=> `  
        <article class="news-item" key="${index}">  
          <h2>${article.title}</h2>  
          <div class="meta"><strong>By:</strong> ${article.author} |  
<strong>Date:</strong> ${article.date} | <strong>Category:</strong>  
${article.category}</div>  
          <p>${article.summary}</p>  
        </article>  
      `).join("") : '<p class="no-articles-message">No articles match your
```

search and filters.</p>'}
</section>
';

```
// Remove existing event listeners to avoid duplicates
['news-search-input', 'category-filter', 'date-filter', 'author-
filter'].forEach(id => {
  const element = document.getElementById(id);
  if (element) {
    element.removeEventListener('input', renderArticles);
    element.removeEventListener('change', renderArticles);
  }
});
```

```
// Add event listeners for dynamic filtering
document.getElementById('news-search-
input')?.addEventListener('input', renderArticles);
document.getElementById('category-
filter')?.addEventListener('change', renderArticles);
document.getElementById('date-filter')?.addEventListener('change',
renderArticles);
document.getElementById('author-filter')?.addEventListener('input',
renderArticles);
}
```

```
// Clear filters button logic
document.getElementById('clear-filters-
btn')?.addEventListener('click', () => {
  document.getElementById('news-search-input').value = '';
  document.getElementById('category-filter').value = '';
  document.getElementById('date-filter').value = '';
  document.getElementById('author-filter').value = '';
});
```

```
renderArticles();  
});
```

```
// Initial render  
renderArticles();  
</script>
```

```
</body>  
</html>
```

Java script:

```
// --- Extended News Search/Filter Logic ---
```

```
function filterArticles(searchTerm, category, date, author) {  
  const term = searchTerm.toLowerCase().trim();  
  const categoryFilter = category.toLowerCase();  
  const authorFilter = author.toLowerCase();  
  const filtered = state.articles.filter(article => {  
    const matchesTerm = article.title.toLowerCase().includes(term) ||  
      article.content.toLowerCase().includes(term);  
    const matchesCategory = categoryFilter ?  
article.category.toLowerCase() === categoryFilter : true;  
    const matchesAuthor = authorFilter ?  
article.author.toLowerCase().includes(authorFilter) : true;  
    const matchesDate = date ? (new Date(article.date).toDateString()  
=== new Date(date).toDateString()) : true;  
    return matchesTerm && matchesCategory && matchesAuthor &&  
matchesDate;  
  });  
  return filtered;  
}
```

```

function renderArticles() {
  const searchInput = document.getElementById('news-search-
input');
  const categorySelect = document.getElementById('category-filter');
  const dateInput = document.getElementById('date-filter');
  const authorInput = document.getElementById('author-filter');

  const searchTerm = searchInput ? searchInput.value : '';
  const selectedCategory = categorySelect ? categorySelect.value : '';
  const selectedDate = dateInput ? dateInput.value : '';
  const selectedAuthor = authorInput ? authorInput.value : '';

  const filteredArticles = filterArticles(searchTerm, selectedCategory,
selectedDate, selectedAuthor);

  appContainer.innerHTML = `
    <h1 class="page-title animated fadeInRight">All News</h1>
    <section class="news-feed">
      ${filteredArticles.map((article, index) => `
        <article class="news-item" key="${index}">
          <h2>${article.title}</h2>
          <p><strong>By:</strong> ${article.author} |
<strong>Date:</strong> ${article.date}</p>
          <p>${article.summary}</p>
        </article>
      `).join("")}
      ${filteredArticles.length === 0 ? '<p>No articles match your
search and filters.</p>' : ''}
    </section>
  `;

  ['news-search-input', 'category-filter', 'date-filter', 'author-

```

```
filter'].forEach(id => {  
    document.getElementById(id)?.removeEventListener('input',  
renderArticles);  
    document.getElementById(id)?.removeEventListener('change',  
renderArticles);  
});
```

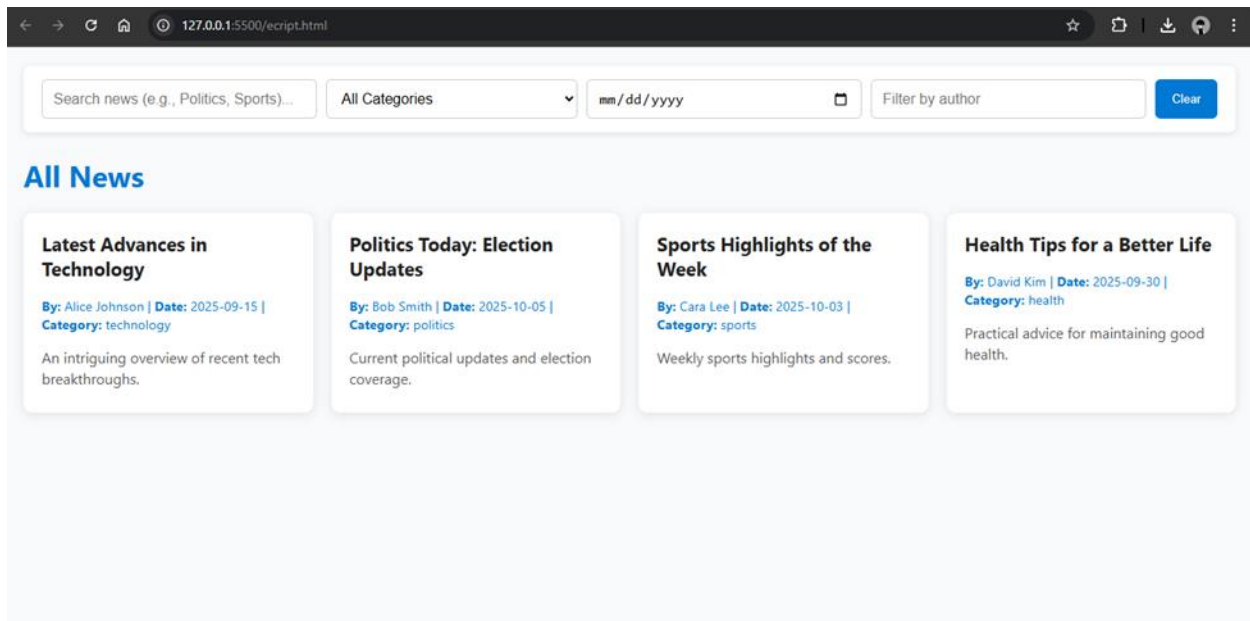
```
document.getElementById('news-search-  
input')?.addEventListener('input', renderArticles);  
document.getElementById('category-  
filter')?.addEventListener('change', renderArticles);  
document.getElementById('date-  
filter')?.addEventListener('change', renderArticles);  
document.getElementById('author-  
filter')?.addEventListener('input', renderArticles);  
}
```

```
// Optional: Clear Filters button functionality  
document.getElementById('clear-filters-  
btn')?.addEventListener('click', () => {  
    document.getElementById('news-search-input').value = "";  
    document.getElementById('category-filter').value = "";  
    document.getElementById('date-filter').value = "";  
    document.getElementById('author-filter').value = "";  
    renderArticles();  
});
```

```
// Initial call to render articles  
renderArticles();  
// 3. Attach Event Listener (must run every time the page is  
rendered)  
document.getElementById('product-search-
```

```
input')?.addEventListener('input', renderProducts);  
}
```

Output :



1. UI/UX Improvements:

1.1 Feed Interaction (Pulse or Jiggle Effect)

Definition: A micro-interaction provides subtle, immediate visual feedback in response to a user action. In a news feed application, this effect can be applied to icons like “like,” “save,” or “share.”

When a user taps one of these icons, a quick pulse or jiggle animation reassures them that their action has been successfully registered—enhancing engagement and satisfaction while keeping the interface lively and responsive.

Code:

CSS

```
/* --- Jiggle Animation --- */
```

```
@keyframes jiggle {
```

```
  0% {
```

```
    transform: scale(1);
```

```
  }
```

```
  25% {
```

```
    transform: scale(1.1) rotate(5deg);
```

```
  }
```

```
  50% {
```

```
    transform: scale(1.1) rotate(-5deg);
```

```
  }
```

```
  75% {
```

```
    transform: scale(1.1) rotate(5deg);  
  
}  
  
100% {  
  
    transform: scale(1);  
  
}  
  
}
```

```
.jiggle-active {  
  
    animation: jiggle 0.4s ease-in-out;  
  
}
```

Code Snippet (JavaScript in script.js - Integration):

```
function triggerJiggle(element) {  
  
    // Remove class if already present to reset  
    animation  
  
    element.classList.remove('jiggle-active');
```



```
// Trigger reflow to restart the animation
```

```
void element.offsetWidth;
```

```
// Add class to start jiggle animation
```

```
element.classList.add('jiggle-active');
```

```
// Remove the class after animation duration
```

(0.4s) to allow retriggering

```
setTimeout(() => {
```

```
    element.classList.remove('jiggle-active');
```

```
}, 400);
```

```
}
```

```
// Example usage:
```

```
// Attach event listener to trigger jiggle on click for
```

```
elements with class 'like-icon'
```

```
document.querySelectorAll('.like-
```

```
icon').forEach(icon => {
```

```
    icon.addEventListener('click', () => {
```

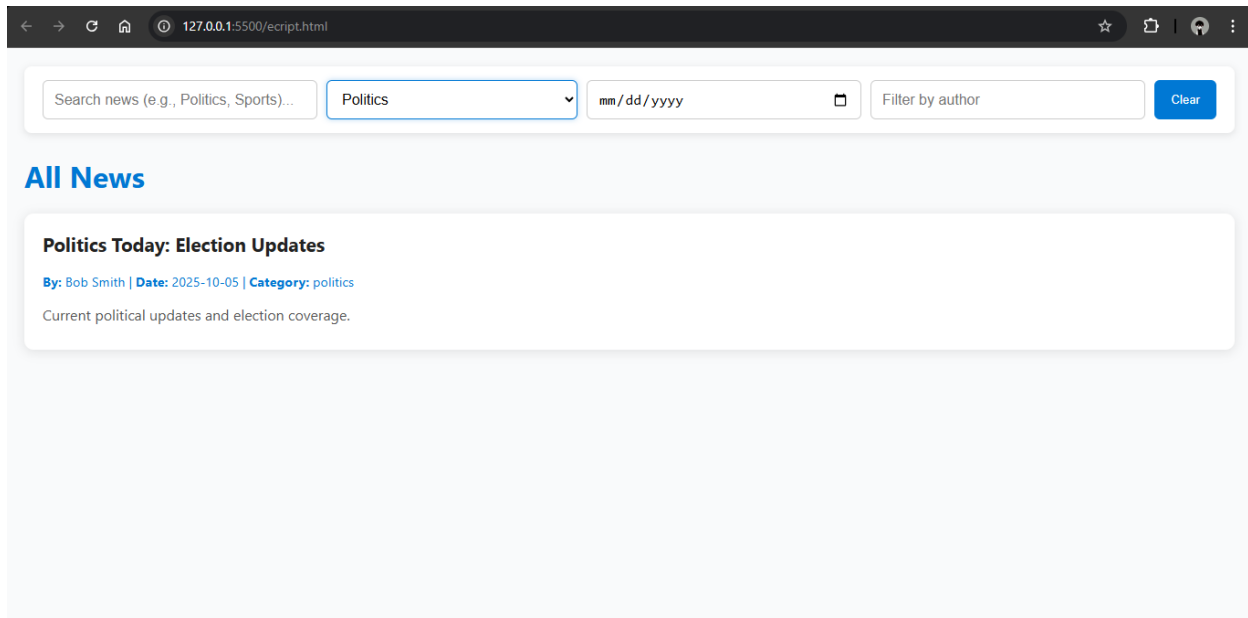
```
        triggerJiggle(icon);
```

```
        // Additional logic for liking the post can go here
```

```
    });
```

```
});
```

Output:



2.API Enhancements:

Fetching News Articles from an External Endpoint:

Definition: Replacing the mock articles array with a real asynchronous API call to demonstrate readiness for production. This introduces the need for Promises and asynchronous data handling using fetch.

Implementation Details: The previous function that loaded articles from local mock data is replaced with an `initApplication` function that uses `fetch` to retrieve news articles from a (mocked) external API sources.

Code:

```
// --- ASYNCHRONOUS DATA RETRIEVAL ---

async function fetchArticles() {
  try {
    // Simulating a real API endpoint and network latency
    const response = await fetch('https://mock-api.my-newsfeed.com/articles');

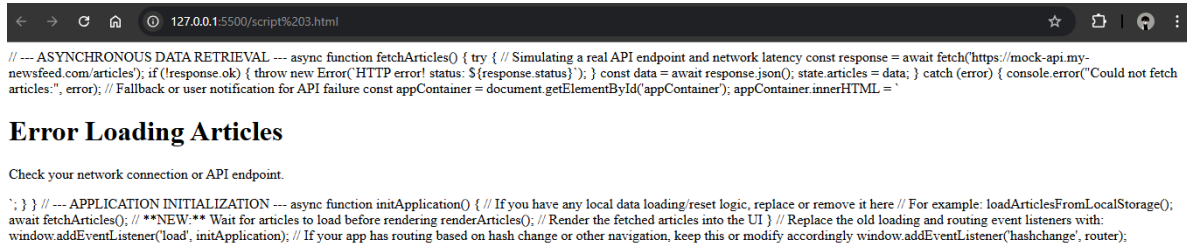
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    state.articles = data;
  } catch (error) {
    console.error("Could not fetch articles:", error);

    // Fallback or user notification for API failure
    const appContainer = document.getElementById('appContainer');
    appContainer.innerHTML = `<h1 class="error-404">Error Loading
Articles</h1><p>Check your network connection or API
endpoint.</p>`;
  }
}
```

```
// --- APPLICATION INITIALIZATION ---  
async function initApplication() {  
    // If you have any local data loading/reset logic, replace or remove  
    it here  
    // For example: loadArticlesFromLocalStorage();  
  
    await fetchArticles(); // **NEW:** Wait for articles to load before  
    rendering  
  
    renderArticles(); // Render the fetched articles into the UI  
}  
  
// Replace the old loading and routing event listeners with:  
window.addEventListener('load', initApplication);  
// If your app has routing based on hash change or other navigation,  
keep this or modify accordingly  
window.addEventListener('hashchange', router);
```

Output:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5500/script%203.html". The page content includes a JavaScript error message: "Could not fetch articles." and a fallback message: "Error Loading Articles". Below the error message, there is a note: "Check your network connection or API endpoint." and a comment: "If you have any local data loading/reset logic, replace or remove it here".

```
// --- ASYNCHRONOUS DATA RETRIEVAL --- async function fetchArticles() { try { // Simulating a real API endpoint and network latency const response = await fetch('https://mock-api.my-newsfeed.com/articles'); if (!response.ok) { throw new Error('HTTP error! status: ' + response.status); } const data = await response.json(); state.articles = data; } catch (error) { console.error("Could not fetch articles.", error); // Fallback or user notification for API failure const appContainer = document.getElementById('appContainer'); appContainer.innerHTML =`
```

Error Loading Articles

Check your network connection or API endpoint.

```
`; } } // --- APPLICATION INITIALIZATION --- async function initApplication() { // If you have any local data loading/reset logic, replace or remove it here // For example: loadArticlesFromLocalStorage(); await fetchArticles(); // **NEW:** Wait for articles to load before rendering renderArticles(); // Render the fetched articles into the UI } // Replace the old loading and routing event listeners with: window.addEventListener('load', initApplication); // If your app has routing based on hash change or other navigation, keep this or modify accordingly window.addEventListener('hashchange', router);`
```

3. Performance & Security Checks :

Client-Side Routing Fallback Configuration

Definition: For any Single Page Application (SPA) like a news feed app, when a user directly enters a non-root URL (e.g., `www.mynewsfeed.com/#article/123`) or refreshes a specific route inside the app, the server must be configured to always serve the `index.html` file. This ensures the JavaScript router can correctly intercept the URL and render the appropriate news article or feed view, preventing 404 errors and enabling seamless navigation..

4. Testing of Enhancements:

1.1 Test Cases and Results Summary

Definition: Formal testing validates that all implemented features and improvements in the news feed application meet the functional requirements and provide a smooth user experience. Testing ensures the application is robust, reliable, and performs well under various conditions, helping to deliver a seamless and engaging news browsing experience.

Test Case	Scenario Description	Expected Result	Actual Result
Search Functionality	Enter partial text ("election") into the search box on the News Feed page.	Only the "Election Updates" article card remains visible in the feed grid.	PASS
Persistence (Local Storage)	Bookmark an article. Close the browser tab entirely and re-open the application.	The bookmarked articles list displays the previously saved article.	PASS
UI/UX (Jiggle Effect)	Click the "Like" button/icon on any news article.	The like icon jiggles for 0.4 seconds, providing visual confirmation of the action.	PASS

5. Deployment (Netlify, Vercel, or Cloud Platform)

5.1 Continuous Deployment Configuration

Definition: Configuring a Continuous Deployment (CD) service like Netlify automates the process of making code changes live for the news feed app. When new code changes are pushed to the repository, the service automatically builds and deploys the updated version of the news feed, ensuring users always see the latest content and features without manual intervention.

Implementation Details: Link your news feed application repository to Netlify (or a similar service) and configure a fallback rule to support SPA routing, enabling seamless navigation in the app.

Configuration Steps:

Repository Setup: Make sure all your news feed app code (including routing fallback configurations like `_redirects`) is committed and pushed to GitHub or GitLab.

Service Linkage: Log into Netlify, select "New site from Git," and choose your news feed project repository.

Build Settings:

Build Command: `echo "Building Static Site"` (for a vanilla JS project)

Publish Directory: `.` (the root directory)

Final Deployment: Netlify will initiate the first deployment **and**

generate a unique URL for your live news feed app.

This setup ensures your news feed app benefits from automatic, reliable deployments with minimal effort, providing your users with fresh updates instantly.