

**COLLEGE CODE : 9604**

**COLLEGE NAME : C.S.I INSTITUTE OF TECHNOLOGY**

**DEPARTMENT : INFORMATION TECHNOLOGY**

**STUDENT NM ID: 7EE32B0131E5363B6FF4198BC0B0A8D9**

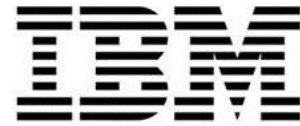
**ROLL NO : 960423205009**

**DATE : 23/09/2025**

**SUBMITTED BY,**

**NAME : JEFFRIN J J**

**MOBILE NO : 7558126173**



## PHASE 3-MVP IMPLEMENTATION

### NEWS FEED APPLICATION

#### Project Setup :

- **Create a dedicated project folder**

Organize it with sub-folders for the main application files, styles, and a sample news-articles database or API configuration.

- **Set up the basic application structure**

Use **HTML** for the layout (header, category filters, and feed section), **CSS** for styling (responsive cards, light/dark mode), and **JavaScript** for interactive features such as fetching and displaying articles.

- **Design the news-feed interface**

Build a scrolling feed that shows one article card per entry with a headline, image, brief summary, source, and a “Read More” link.

- **Implement JavaScript functions**

Handle API calls or local data storage, dynamically render article cards, manage category or keyword filters, and support features like infinite scrolling or pagination.

- **Add extra features**

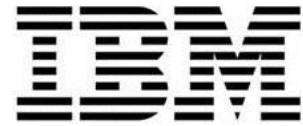
Include live refresh to pull the latest stories, user actions like “bookmark” or “like,” and optional personalization such as saving preferred categories.

- **Provide a highlights or results section**

Offer a “Top Stories” or “Trending” area that summarizes the most recent or most-read articles for quick access.

- **Test thoroughly**

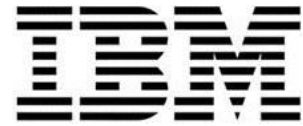
Check for network errors, empty results, and responsiveness across devices to ensure a smooth, engaging reading experience.



## CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>News Feed App</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <header class="header">
    <h1>Daily News Feed</h1>
    <nav class="filters">
      <button data-category="all" class="filter-btn active">All</button>
      <button data-category="world" class="filter-btn">World</button>
      <button data-category="technology" class="filter-
btn">Technology</button>
      <button data-category="sports" class="filter-btn">Sports</button>
      <button data-category="entertainment" class="filter-
btn">Entertainment</button>
    </nav>
  </header>

  <main class="news-container">
    <!-- Articles will be injected here dynamically -->
    <div id="news-list" class="news-list">
```



<!-- Example of a single article card -->

<!--

<article class="news-card">



<div class="news-content">

<h2 class="news-title">Headline goes here</h2>

<p class="news-summary">Short description of the news story.</p>

<a href="#" class="read-more" target="\_blank">Read More</a>

</div>

</article>

-->

</div>

<div class="loading" id="loading">Loading latest articles...</div>

</main>

<footer class="footer">

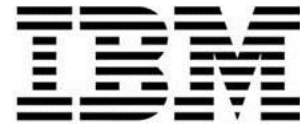
<p>&copy; 2025 News Feed MVP</p>

</footer>

<script src="script.js"></script>

</body>

</html>



## OUTPUT :

Daily News Feed

[All] [World] [Technology] [Sports] [Entertainment]

Loading latest articles...

© 2025 News Feed MVP

## CORE FEATURES IMPLEMENTATION :

- **Dynamic News Feed Interface**

Display articles as interactive cards with headline, image, summary, and a “Read More” link. Cards load automatically from an API or local data file.

- **Live Updates / Auto-Refresh**

Periodically fetch the latest stories so the feed stays current without manual reload.

- **Category & Keyword Filtering**

Allow users to filter by topics (e.g., World, Technology, Sports) or search for specific keywords.

- **User Engagement & Bookmarks**

Enable actions like “Like,” “Bookmark,” or “Save for Later,” with the state stored in local storage or a lightweight database.

- **Trending / Top Stories Section**

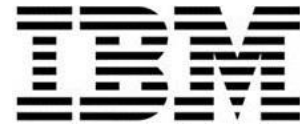
Highlight the most-read or most-liked articles for quick discovery.

- **Pagination or Infinite Scroll**

Support smooth navigation through large numbers of articles with “Load More” buttons or automatic infinite scrolling.

- **Responsive Design**

Ensure the layout adapts seamlessly to desktop, tablet, and mobile screens.



- **Error Handling & Offline Support**

Provide clear messages if the API fails or the user is offline, and optionally cache recent articles for offline reading.

## Basic Implementation Overview :

### 1. News Feed Interface

- Display a list of news articles as interactive cards showing **headline, image, brief summary, source, and “Read More” link**.
  - Highlight an article card when hovered or selected.
  - Allow category filters (e.g., *World, Technology, Sports*) and keyword search.
- 

### 2. Live Updates / Refresh

- Implement a **refresh timer** or background polling to fetch the latest articles at regular intervals.
  - Auto-insert new stories at the top of the feed when updates arrive.
- 

### 3. User Interaction & Tracking

- Maintain a **bookmark or “liked” list** so users can save articles.
  - Optionally track read/unread status and update the UI in real time.
- 

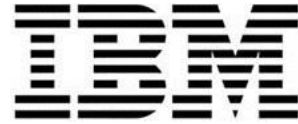
### 4. Navigation & Flow

- Provide **infinite scrolling** or a **“Load More” button** to seamlessly load additional articles.
  - Smoothly handle transitions when users switch categories or perform a new search.
- 

### 5. Highlights / Results Section

- Offer a **“Trending” or “Top Stories”** panel that summarizes the most-read or most-liked articles.
- Show a friendly message or fallback view if no articles are available (e.g., API failure or empty search).

**CODE :**



```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h3>News: Paris Updates</h3>
```

```
<p>Paris has recently reopened several museums and tourist  
spots after renovations.</p>
```

```
<button onclick="bookmark('Paris News')">Bookmark</button>
```

```
<p id="feedback"></p>
```

```
<script>
```

```
function bookmark(articleTitle){
```

```
    // Display feedback when user bookmarks an article
```

```
    document.getElementById("feedback").textContent =  
    `"${articleTitle}" added to your bookmarks!`;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

**OUTPUT:**

News: Paris Updates

Paris has recently reopened several museums and tourist spots after renovations.

[ Bookmark ]

"Paris News" added to your bookmarks!

## DATA STORAGE ( LOCAL STATE / DATABASE ) :

### Local State / Local Storage

- Useful for storing **user preferences, bookmarked articles, read/unread status, and temporary app state**.
  - Options include `localStorage` for web apps (simple key-value storage, up to ~5–10MB) or `AsyncStorage` for React Native.
  - Not suitable for **large datasets or complex queries** like hundreds of full news articles.
  - Ideal for **caching recently read articles, storing filter settings, or keeping user session data**.
- 

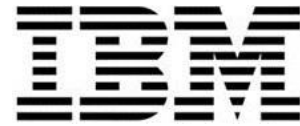
### Local Database

- For storing **large news article sets or offline content**, a lightweight local database like **SQLite** is recommended.
  - Databases like **Realm** or **Isar** provide offline support, syncing, and fast queries.
  - **IndexedDB** is a powerful local database for web apps, allowing storage of structured news data beyond `localStorage` limits.
  - A local database allows **efficient search, category filtering, and sorting of hundreds or thousands of articles**.
  - Keeping content fresh may require **syncing with a remote service** (e.g., Firebase, Supabase, or a custom backend).
- 

### Hybrid Approach

- Store the **main news dataset** in SQLite or another local database for fast access.
- Use `localStorage` for **transient user data** such as bookmarks, read/unread status, and last-read position.
- Sync the database periodically with the cloud to provide **updated articles** without requiring a full app reinstall.





## TESTING CORE FEATURES :

### Functional Testing

- Verify that the news feed loads correctly, with articles appearing as expected.
  - Check that user interactions are handled properly (e.g., clicking “Read More,” bookmarking articles, switching categories).
  - Validate dynamic updates like auto-refresh or infinite scroll.
  - Test core flows: opening the app, navigating between categories, applying search filters, and loading additional articles.
  - Ensure optional features like user login, profile preferences, or saved bookmarks work correctly.
- 

### User Interface and Usability Testing

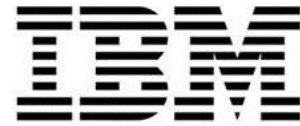
- Test UI elements such as buttons, filter controls, cards, and interactive components for responsiveness.
  - Check layout consistency across devices, screen sizes, and orientations.
  - Simulate real user gestures like tap, swipe, and scroll on mobile devices.
- 

### Compatibility Testing

- Test across different operating systems, browser versions, and device types.
  - Validate performance under varied network conditions (slow connections, offline mode).
- 

### Automated Testing

- Use frameworks like **Selenium** or **Appium** to automate repetitive regression tests.
  - Run tests on multiple devices simultaneously using cloud platforms like **BrowserStack** or **Sauce Labs**.
-



## Beta Testing

- Release the app to a limited audience to capture real-world usage feedback.
  - Identify device-specific or environment-related issues that were missed during controlled testing.
- 

## Performance Testing

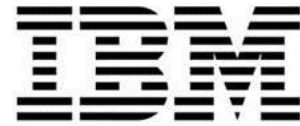
- Ensure the app handles large news datasets efficiently and loads articles quickly.
  - Monitor app behavior under concurrent users or high interaction (e.g., scrolling, filtering, bookmarking).
- 

## Error and Edge Case Testing

- Check app behavior with invalid inputs, empty searches, or missing article data.
  - Validate offline functionality, data caching, and recovery after crashes or network interruptions.
- 

## Debugging and Logs

- Use logs, analytics, and crash reports to investigate issues and improve stability.
- Combining these strategies ensures the core features of the **interactive News Feed App** are thoroughly tested for a smooth, cross-platform user experience



## VERSION CONTROL ( GITHUB ) :

### Purpose

1. **Track changes** – GitHub allows you to monitor every change made to your news feed application code.
  2. **Collaboration** – Multiple developers can work on different features simultaneously (e.g., adding bookmarks, implementing category filters).
  3. **Backup** – Your app code is safely stored in a remote repository, reducing the risk of data loss.
- 

### GitHub Features

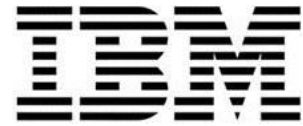
1. **Repositories** – Store and manage your News Feed app project in a repository.
  2. **Commits** – Track changes to the codebase with commits, documenting new features like infinite scroll or API integration.
  3. **Branches** – Work on separate branches for new features or bug fixes without affecting the main version.
  4. **Pull Requests** – Review and merge changes from contributors, ensuring quality and consistency.
- 

### Best Practices

1. **Regular commits** – Make frequent commits to capture progress, especially when implementing critical features like offline caching or real-time updates.
  2. **Clear commit messages** – Write descriptive commit messages (e.g., “Add category filter and bookmark feature”).
  3. **Branching strategy** – Use a branching strategy (e.g., `main` for stable code, `develop` for ongoing work, feature branches for individual features).
- 

### GitHub Tools

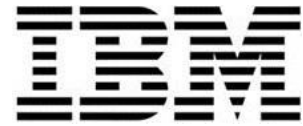
1. **Git** – Manage your local repository and synchronize changes with GitHub.
2. **GitHub Desktop** – Graphical interface for managing commits, branches, and pull requests.
3. **GitHub CLI** – Command-line tool for advanced GitHub interactions



**CODE :**

```
# short_news_feed.py
```

```
articles = {  
    "Paris reopens museums after renovations":  
    "https://news.example.com/paris-museums",  
    "Tech giant releases new AI tool":  
    "https://news.example.com/ai-tool"  
}  
  
bookmarked = []  
  
print("=== Daily News Feed ===\n")  
  
for title, link in articles.items():  
    print(f"Headline: {title}")  
    print(f"Read more: {link}")  
    choice = input("Bookmark this article? (yes/no): ").strip().lower()  
    if choice == "yes":  
        bookmarked.append(title)  
        print("✅ Bookmarked!\n")  
    else:  
        print("Skipped.\n")  
  
print("=== Your Bookmarked Articles ===")
```



if bookmarked:

```
for idx, article in enumerate(bookmarked, 1):
```

```
    print(f"{idx}. {article}")
```

else:

```
    print("No articles bookmarked.")
```

### OUTPUT:

=== Daily News Feed ===

Headline: Paris reopens  
museums after renovations

Read more:

[https://news.example.com/  
paris-museums](https://news.example.com/paris-museums)

Bookmark this article?

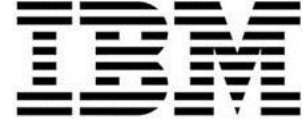
(yes/no): no

Skipped.

Headline: Tech giant  
releases new AI tool

Read more:

[https://news.example.com/  
ai-tool](https://news.example.com/ai-tool)



Bookmark this article?

(yes/no): no

Skipped.

=== Your Bookmarked

Articles ===

No articles bookmarked.