

# **NEWSHUB: A GLOBAL AND LOCAL NEWS AGGREGATION PLATFORM**

**Integrative Programming and Technologies**

In Partial Fulfillment of the Requirements for the Degree Bachelor of Science in  
Information Technology

By

**Candinato, Jomel** *(Main Programmer)*

**Medalla, Lawrence**

**Novero, Jeremy**

**Valencia, Paul Dexter**

December 2025

# Project Contents

<b>Cover Page</b>	1
<b>I. Website Description</b>	3
<b>II. Website Design</b>	4
a. Home Page ( <i>Guest User</i> )	4
b. Login Page	4
c. Register Page	5
d. Forgot Password Page	5
e. Home Page ( <i>With Logged in User</i> )	6
f. News Detail Page	6
g. About Page	7
<b>III. Framework Used</b>	7
<b>IV. API Integration</b>	8
<b>V. UI Components</b>	9
a. Navbar	9
b. Country Toggle	9
c. Category Filter	9
d. News List	10
e. News Card	10
f. News Details	10
g. Add to Favorites	10
h. Refresh News Button	10
i. Footer	10
<b>VI. Event-Driven Interactions</b>	10
1. Search Functionality	10
2. Category Selection	11
3. Country Toggle	11
4. Add to Favorites	12
5. Article Reading	12
<b>VII. Try/Catch Error Handling</b>	13
1. Login Error Handling	13
2. Registration Error Handling	14
3. Search Error Handling	14
4. Article Fetch Error Handling	15
5. API Error Handling	16
<b>VIII. SOURCE CODE</b>	16

## I. Website Description

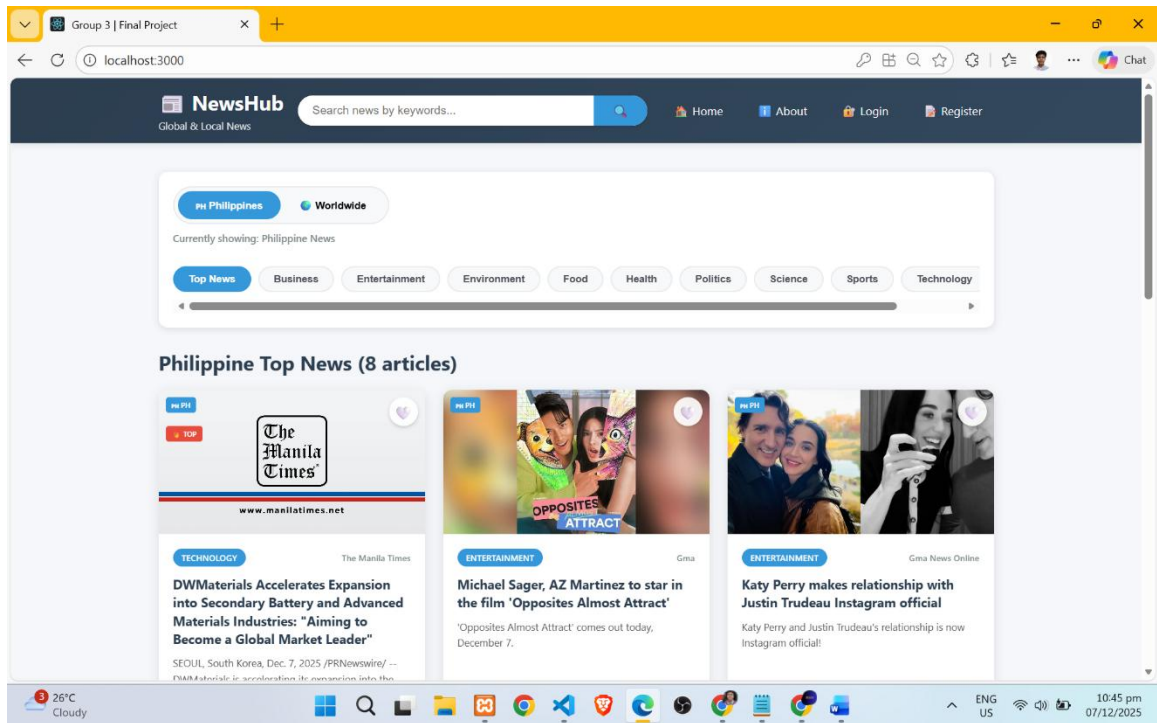
**NewsHub** is a modern and responsive news aggregation platform, offering users real-time access to both worldwide and Philippine-specific news articles. Developed as a final project for Integrative Programming & Technologies, the platform serves as a comprehensive digital news source that unites global and local headlines.

The platform features an intuitive user interface supported by robust filtering capabilities. This allows users to browse news by key categories such as Business, Technology, Sports, Politics, Entertainment, Environment, Food, Health, Science, and World News. Users can seamlessly toggle between Philippine and international news sources, ensuring they stay informed about both local and global events. Furthermore, the platform includes advanced search functionality that enables keyword-based news discovery across all available categories and sources.

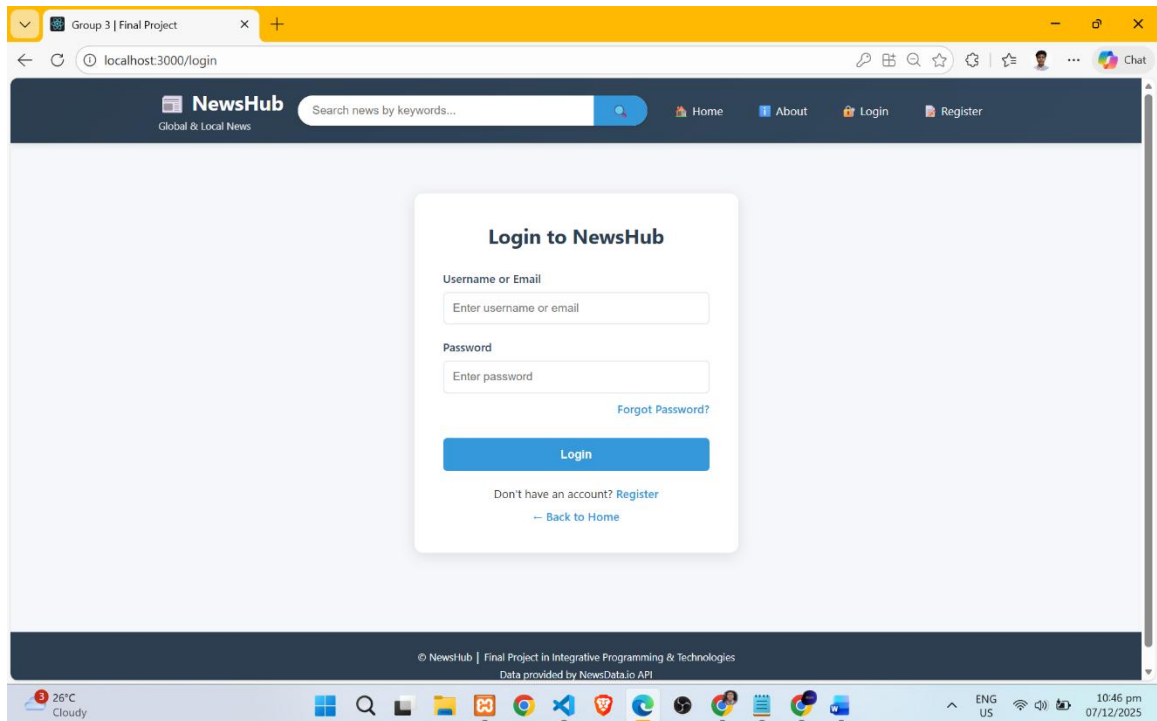
NewsHub employs a clean, card-based layout for news presentation, with each article displaying key information, including headlines, summaries, publication dates, sources, and relevant images. The platform provides detailed article views with related news suggestions, creating an engaging reading experience. Designed with performance and user experience in mind, the platform includes loading indicators, error-handling mechanisms, and a responsive design that works across desktops, laptops, and mobile devices. By leveraging the NewsData.io API, NewsHub ensures access to current, reliable news content while offering users practical experience in full-stack web development with React, API integration, and modern JavaScript practices.

## II. Website Design

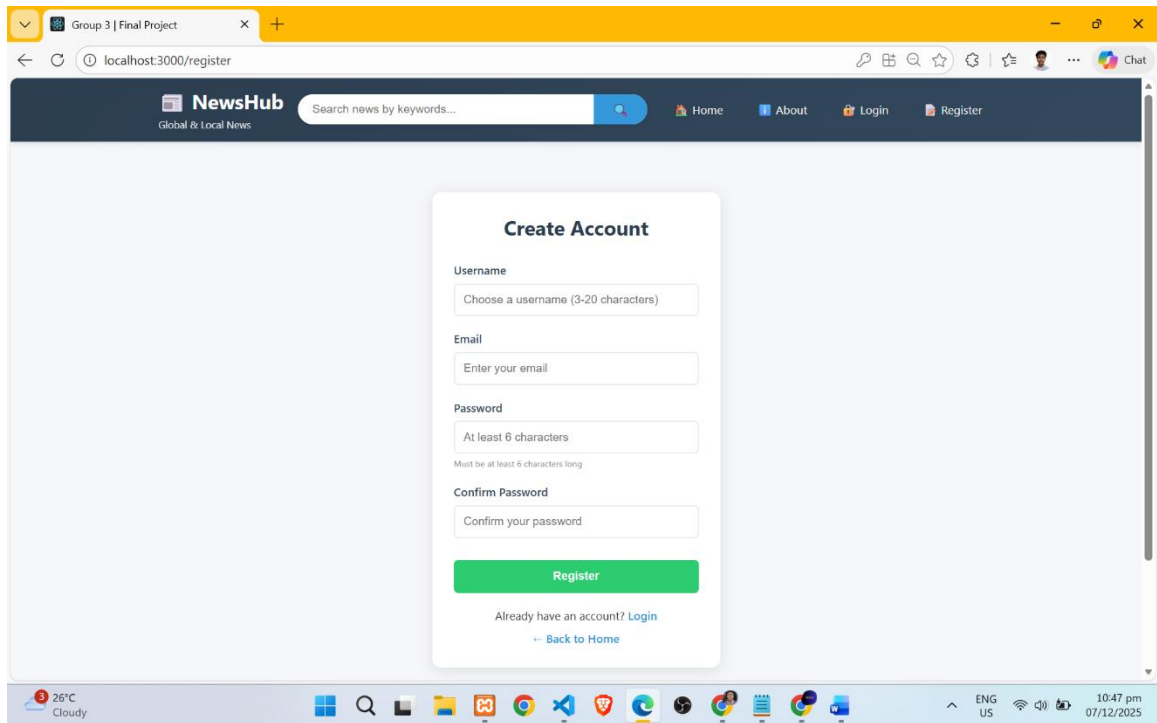
### a. Home Page (Guest User or Logged Out):



### b. Login Page:

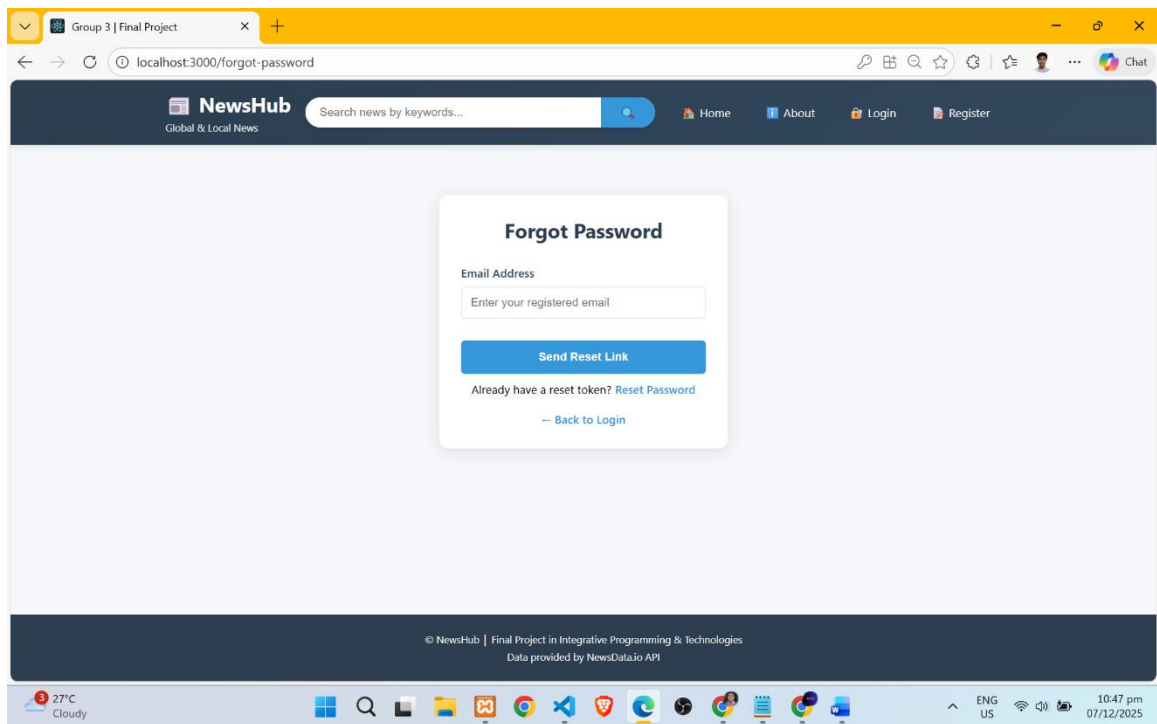


### c. Register Page:



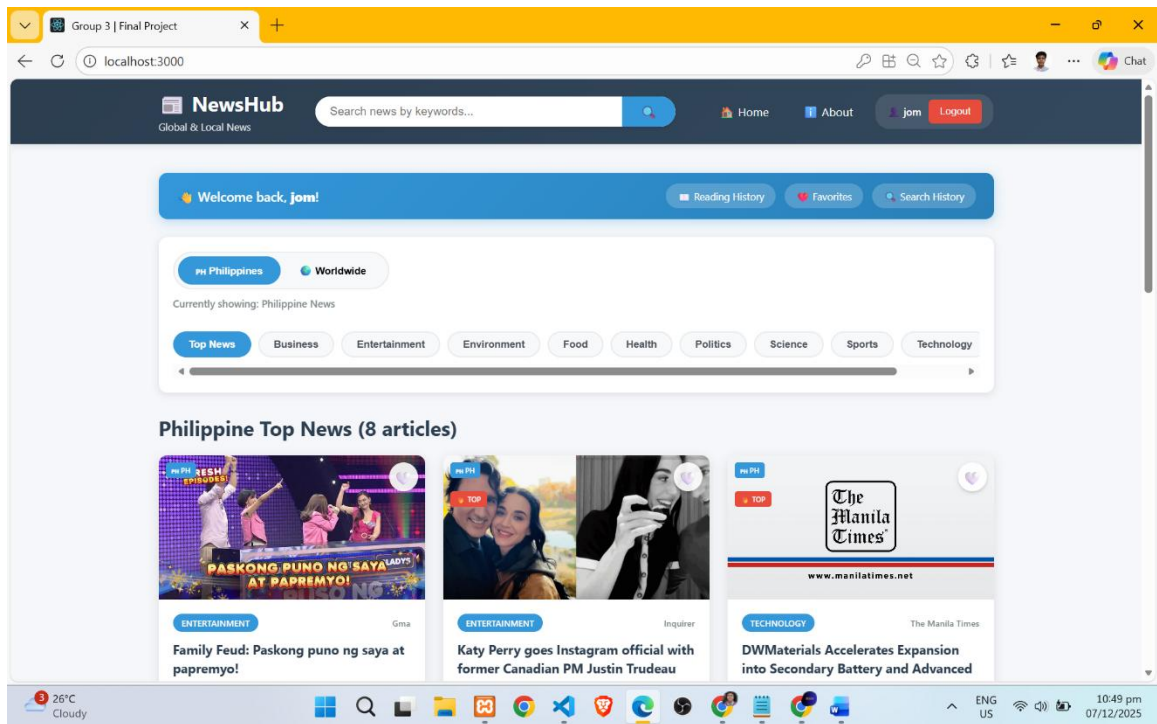
The screenshot shows a web browser window with the URL `localhost:3000/register`. The page features a dark blue header with the NewsHub logo, a search bar, and navigation links for Home, About, Login, and Register. The main content area is light blue and contains a white 'Create Account' form. The form has four input fields: Username (with a hint 'Choose a username (3-20 characters)'), Email (with a hint 'Enter your email'), Password (with a hint 'At least 6 characters' and a note 'Must be at least 6 characters long'), and Confirm Password (with a hint 'Confirm your password'). A green 'Register' button is at the bottom of the form. Below the button, there is a link 'Already have an account? Login' and a link 'Back to Home'.

### d. Forgot Password Page:

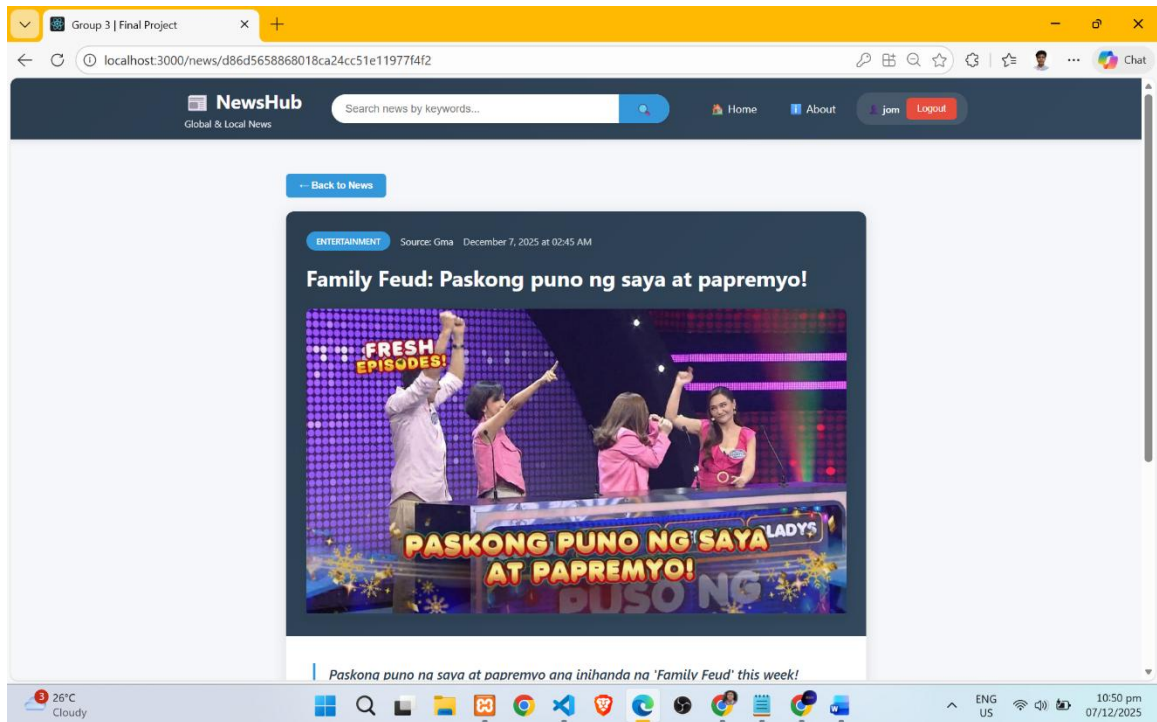


The screenshot shows a web browser window with the URL `localhost:3000/forgot-password`. The page features a dark blue header with the NewsHub logo, a search bar, and navigation links for Home, About, Login, and Register. The main content area is light blue and contains a white 'Forgot Password' form. The form has one input field: Email Address (with a hint 'Enter your registered email'). A blue 'Send Reset Link' button is at the bottom of the form. Below the button, there is a link 'Already have a reset token? Reset Password' and a link 'Back to Login'.

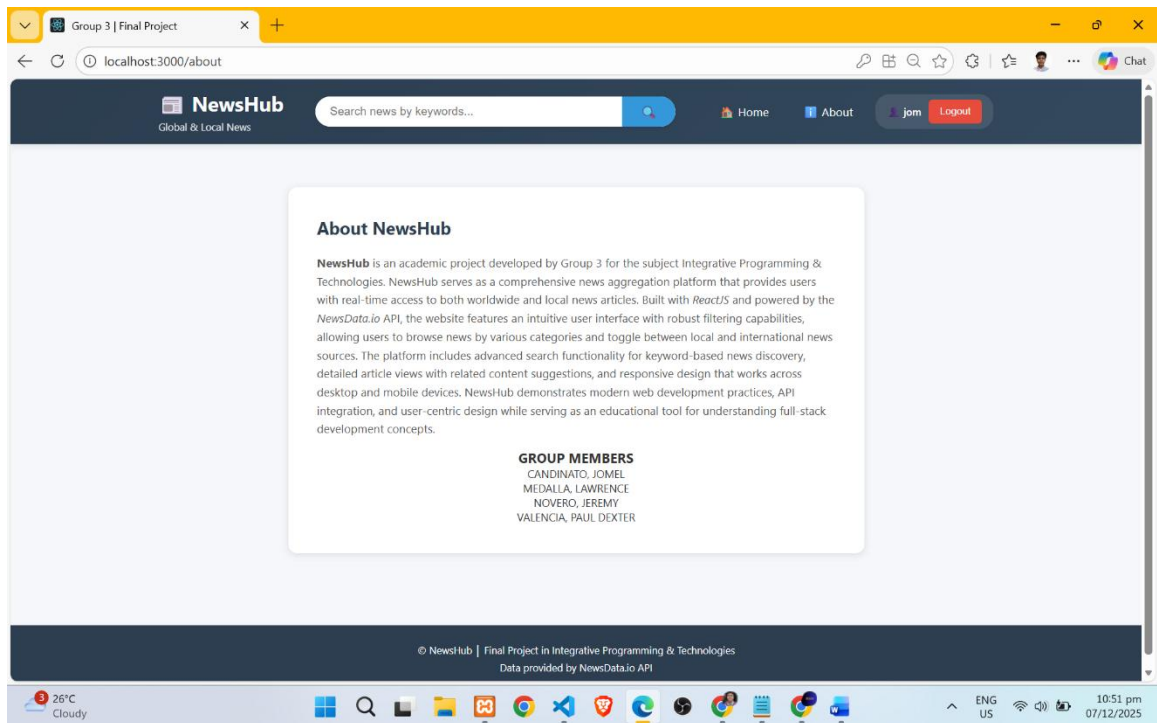
## e. Home Page: (Logged In)



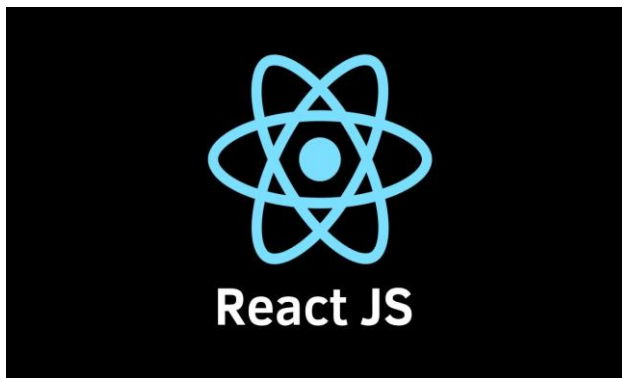
## f. News Detail Page:



## g. About Page:



## III. Framework Used



NewsHub uses **ReactJS**, a declarative, component-based JavaScript library for building user interfaces, developed by Facebook. NewsHub utilizes React's latest features including functional components with Hooks (`useState`, `useEffect`, `useCallback`), React Router for client-side routing, and modern state management patterns. The application follows React's component architecture for reusable UI elements and efficient rendering through the Virtual DOM.

## IV. API Integration

*newshub/src/services/api.js:*

```
import axios from 'axios';

const API_KEY = process.env.REACT_APP_NEWS_API_KEY || 'pub_0755496150ae44728c2f6498fa4172b5';
const BASE_URL = 'https://newsdata.io/api/1';

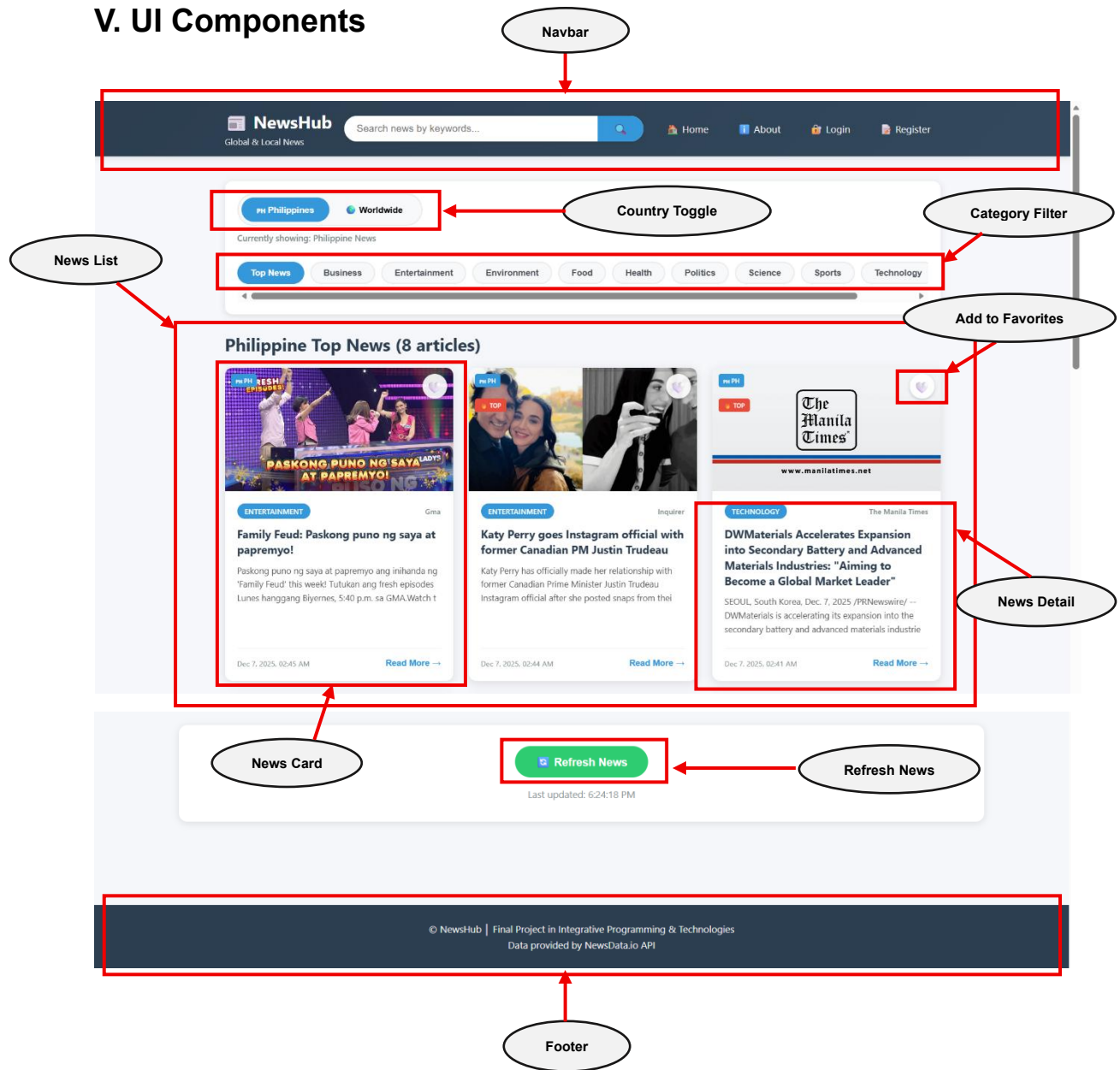
class NewsAPI {
  constructor() {
    this.api = axios.create({
      baseURL: BASE_URL,
      params: {
        apikey: API_KEY,
      },
    });
  }
}
```

```
async getLatestNews(country = 'ph', category = null) {
  try {
    const params = {
      country: country === 'ph' ? 'ph' : 'us,gb,au,in',
      language: 'en',
    };
  }
}
```

NewsHub seamlessly integrates with the [NewsData.io](https://newsdata.io) API to fetch real-time news articles from various sources worldwide. This integration is handled through a dedicated API service layer (NewsAPI class) that manages API requests, parameter formatting, and response processing. The system dynamically constructs API calls based on user selections (country, category, search queries) and includes comprehensive error handling for network issues, API limits, and data validation. When the primary API fails or returns no results, the application gracefully falls back to mock data, ensuring users always have content to view. The integration supports both Philippine-specific news filtering and worldwide news aggregation, with automatic filtering of irrelevant sources for better user experience. API keys are securely managed through environment variables, and the system implements rate limiting awareness and error recovery mechanisms to maintain reliable service even during API disruptions.



## V. UI Components



- Navbar:** The navigation bar at the top of the page includes the website name and logo, a search bar for keyword-based news discovery, and navigation links. For authenticated users, it displays user-specific features like Reading History, Favorites, and Search History, along with a logout button.
- Country Toggle:** A component with two buttons to switch between Local and Global news.
- Category Filter:** A component that renders a list of category buttons.

- d. **News List:** Displays a list of news articles.
- e. **News Card:** Displays a single news article with an image, category, source, title, description, and a "Read More" link.
- f. **News Details:** A detailed view of a single news article. When a user clicks on "Read More", it will show the full article content, related articles, and a link to the original source.
- g. **Add to Favorites:** A heart icon button on each News Card that lets users bookmark articles.
- h. **Refresh News button:** Allows users to manually refresh the current news feed.
- i. **Footer:** Contains copyright information, project attribution, and API credit.

## VI. Event-Driven Interactions

### 1. Search Functionality

*newshub/src/components/NewsList.js*

```
const performSearch = async (query) => {  
  try {  
    setSearchLoading(true);  
    setError(null);  
    setLastSearchQuery(query);  
    const data = await NewsAPI.searchNews(query, country);  
  
    if (data.results && data.results.length > 0) {  
      setSearchResults(filteredResults);  
      saveSearchHistory(query, filteredResults.length);  
    }  
  } catch (err) {  
    console.error('Error in performSearch:', err);  
    setError('No results found. Try a different search term.');
```

When a user submits a search query via the navbar search box, the `performSearch` function is called. It sends the query to the News API, filters results based on country preferences, saves the search to history (if logged in), and displays the results in a dedicated section.

## 2. Category Selection

*newshub/src/components/NewsList.js*

```
const handleCategoryChange = useCallback((selectedCategory) => {  
  setCategory(selectedCategory);  
  setSearchResults(null);  
  fetchNews(country, selectedCategory);  
}, [country, fetchNews]);
```

Clicking any category button triggers the `handleCategoryChange` function, which updates the active category state, clears any existing search results, and fetches new articles specific to that category from the current country's news sources.

## 3. Country Toggle

*newshub/src/components/NewsList.js*

```
const handleCountryToggle = useCallback((selectedCountry) => {  
  setCountry(selectedCountry);  
  // If there's an active search, this will keep search results but update country context  
  if (searchResults) {  
    // Re-fetch search results for the new country  
    if (lastSearchQuery) {  
      const performSearch = async () => {  
  
        performSearch();  
      }  
    } else {  
      fetchNews(selectedCountry, category);  
    }  
  }  
});
```

When a user clicks either the "Philippines" or "Worldwide" button, the `handleCountryToggle` function is triggered. This updates the country state, fetches new articles from the API based on the selected country, and updates the UI accordingly. If a search was active, it re-executes the search with the new country parameter.

## 4. Add to Favorites

*newshub/src/components/NewsCard.js*

```
const handleFavoriteClick = async (e) => {
  e.stopPropagation();

  if (!user) {
    alert('Please login or register an account first to add favorites.');
```

```
    navigate('/login');
    return;
  }

  const response = await fetch('http://localhost:5000/api/favorites', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${token}`
    },
    body: JSON.stringify({
      article_id: getArticleId(),
      article_title: article.title,
      article_category: getCategory(),
      article_source: getSource(),
      article_image: getImageUrl(),
      article_url: article.link || article.url || '#',
      description: article.description || '',
      pubDate: article.pubDate || article.publishedAt || new Date().toISOString()
    })
  });

  setIsFavorite(data.isFavorite);
```

Clicking the heart icon on any news card triggers the `handleFavoriteClick` function. It checks authentication first, then toggles the article's favorite status by making a POST request to the backend API. The UI immediately updates to reflect the new state with visual feedback.

## 5. Article Reading

*newshub/src/components/NewsCard.js*

```
<Link
  to={` /news/${encodeURIComponent(article.article_id || article.title)}`}
  style={styles.readMore}
>
  Read More →
</Link>
```

*newshub/src/components/NewsDetail.js*

```
<button onClick={handleBack} style={styles.backButton}>  
  ← Back to News  
</button>
```

Article navigation in NewsHub utilizes React Router's navigation events to create a smooth, single-page application experience. Clicking the *'Read More'* link on any news card triggers an event that encodes the article title into the URL and navigates to the detailed view, maintaining application state without page reloads. The *'Back to News'* button in the detail view similarly uses event-driven navigation to return users to their previous context. These navigation events preserve browsing history and state, allowing users to explore articles deeply while maintaining their place in news listings, creating an intuitive reading flow reminiscent of native applications.

## VII. Try/Catch Error Handling

### 1. Login Error Handling

*newshub/src/components/Login.js*

```
try {  
  console.log('Attempting login with:', { username, password });  
  const result = await login(username, password);  
  console.log('Login result:', result);  
  
  if (result.success) {  
    console.log('Login successful');  
    navigate('/');  
  } else {  
    console.error('Login failed:', result.error);  
    setError(result.error);  
  }  
} catch (err) {  
  console.error('Login error:', err);  
  setError('An error occurred. Please try again.');
```

```
} finally {  
  setLoading(false);  
}
```

During login attempts, the system validates credentials against the backend database. If authentication fails due to invalid credentials, network issues, or server errors, appropriate error messages guide users toward resolution while protecting account security.

## 2. Registration Error Handling

*newshub/src/components/Register.js*

```
try {
  const result = await register(username, email, password);
  if (result.success) {
    navigate('/');
  } else {
    setError(result.error);
  }
} catch (err) {
  setError('An error occurred. Please try again.');
```

Registration includes comprehensive validation for username availability, email format, password strength, and confirmation matching. Each validation failure provides specific guidance, while backend errors are caught and presented in a non-technical manner to facilitate user correction.

## 3. Search Error Handling

*newshub/src/components/NewsList.js:*

```
try {
  setSearchLoading(true);
  setError(null);
  setLastSearchQuery(query);
  const data = await NewsAPI.searchNews(query, country);

  if (data.results && data.results.length > 0) {
    setSearchResults(data.results);
  } else {
    throw new Error('No search results found');
  }
} catch (err) {
  console.error('Error in performSearch:', err);
  setError('No results found. Try a different search term.');
```

This specific error handling addresses the scenario when users search with invalid or non-matching keywords. When the API call returns either no results or an error response, the *catch* block is triggered, logging the technical error for debugging purposes while displaying the user-friendly message: *"No results found. Try a different search term."* The system also sets *searchResults* to an empty array to ensure the interface shows zero results clearly, rather than

maintaining stale data or showing a broken state. This approach provides immediate, constructive feedback to users, guiding them to refine their search terms while maintaining application stability.

#### 4. Article Fetch Error Handling

*newshub/src/components/NewsDetail.js:*

```
try {
  setLoading(true);
  setError(null);

  const data = await NewsAPI.getLatestNews();

  if (data.results && data.results.length > 0) {
    // Find article by title (encoded in URL)
    const decodedId = decodeURIComponent(id);
    const foundArticle = data.results.find(
      (article) =>
        article.article_id === decodedId ||
        article.title === decodedId ||
        article.link?.includes(decodedId)
    );

    if (foundArticle) {
      setArticle(foundArticle);
      // Get related articles (same category)
      const related = data.results
        .filter(a => a.category?.[0] === foundArticle.category?.[0] && a.article_id !==
          foundArticle.article_id)
        .slice(0, 3);
      setRelatedArticles(related);
    } else {
      throw new Error('Article not found');
    }
  } else {
    throw new Error('No articles available');
  }
} catch (err) {
  console.error('Error fetching article:', err);
  setError('Article not found. It might have been removed or the link is incorrect.');
  // Fallback to mock data
  setArticle(getMockArticle());
} finally {
  setLoading(false);
}
```

This detailed error handling addresses the complex scenario of fetching individual articles in a news aggregation system. Since the API lacks a direct endpoint for single articles, the *try* block implements a workaround by fetching all news and filtering for the requested article. The error handling accounts for multiple failure points: network errors, empty API responses, and missing articles. When failures occur, the *catch* block provides specific guidance about why an article might be unavailable while implementing a fallback to mock data, ensuring users always receive some content rather than encountering blank pages or broken links.

## 5. API Error Handling

*newshub/src/components/NewsList.js:*

```
try {  
  
  if (data.results && data.results.length > 0) {  
  
    setArticles(filteredArticles);  
    setSearchResults(null);  
  } else {  
    console.log('No articles found from API, using mock data');  
    // Use mock data  
    const mockData = getMockData(selectedCountry, selectedCategory);  
    setArticles(mockData);  
  }  
} catch (err) {  
  console.error('Error in fetchNews:', err);  
  setError(`Failed to load ${selectedCategory} news. Please try again later.`);  
  
  // Use mock data  
  const mockData = getMockData(selectedCountry, selectedCategory);  
  setArticles(mockData);  
}
```

When the [NewsData.io](https://newsdata.io) API encounters issues (expired API key, rate limits, network problems), the system gracefully falls back to mock data, ensuring users always have content to view while displaying appropriate warnings about the temporary data source.