**Tagging for posts**

General Idea:

**Fetching Articles**  
Your aggregator retrieves article data (title, content, publication date).

**Automatic Categorization (Backend):**

* Each article's title and content are combined into one string.
* Keyword-based categorization function matches this combined string against predefined keyword lists.
* If matches occur, assign the matched categories to the article; if not, assign it to "Uncategorized".

**Store in Database:**

* Save each article along with its category tags.

**Frontend Integration (Dashboard):**

* Display each article with its assigned categories.
* For articles categorized as "Uncategorized", provide a dropdown menu allowing users to select an appropriate category manually.

**Manual Category Updating:**

* When users select a new category, the frontend sends a request to your backend.
* Your backend updates the article's stored category in the database accordingly.

General Idea of the code – This will change when actually trying to integrate it with our dashboard working code and integrate it with the aggregator

Step 1:

Make a file called categories.js to define your keyword-based categories clearly:

// categories.js

export const categoryKeywords = {

Technology: ["tech", "software", "hardware", "programming", "gadgets"],

Sports: ["sports", "soccer", "basketball", "tennis"],

Politics: ["election", "government", "politics", "policy", "president"],

Entertainment: ["movies", "music", "celebrity", "television"],

Business: ["economy", "stocks", "finance", "market", "business"],

Health: ["health", "medicine", "fitness", "nutrition"],

};

Step 2:

Use these keywords to automatically assign categories to articles clearly:

// categorizeArticle.js

import { categoryKeywords } from "./categories";

export const categorizeArticle = (article) => {

const matchedCategories = new Set();

const content = `${article.title} ${article.content}`.toLowerCase();

Object.entries(categoryKeywords).forEach(([category, keywords]) => {

keywords.forEach(keyword => {

if (content.includes(keyword.toLowerCase())) {

matchedCategories.add(category);

}

});

});

// If no category matches, assign "Uncategorized"

return matchedCategories.size > 0

? Array.from(matchedCategories)

: ["Uncategorized"];

};

Step 3: When you pull articles from your aggregator, categorize them automatically:

import { categorizeArticle } from "./categorizeArticle";

async function fetchAndCategorizeArticles() {

const response = await fetch("aggregator-endpoint");

const articles = await response.json();

const categorizedArticles = articles.map(article => ({

...article,

categories: categorizeArticle(article),

}));

// Save categorized articles to your database

await saveArticlesToDatabase(categorizedArticles);

};

Step 4: When showing articles on your dashboard, clearly identify uncategorized articles and let users manually set categories:

// ArticleDashboard.jsx

import { useEffect, useState } from "react";

const categories = ["Technology", "Sports", "Politics", "Entertainment", "Business", "Health"];

export default function ArticleDashboard() {

const [articles, setArticles] = useState([]);

useEffect(() => {

fetch('/api/articles')

.then(res => res.json())

.then(data => setArticles(data));

}, []);

const handleCategoryChange = (id, newCategory) => {

setArticles(articles.map(article =>

article.id === id ? { ...article, categories: [newCategory] } : article

));

fetch(`/api/articles/${id}/update-category`, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({ category: newCategory }),

});

};

return (

<div>

{articles.map(article => (

<div key={article.id}>

<h3>{article.title}</h3>

<p>{article.content}</p>

<div><strong>Category:</strong> {article.categories.join(", ")}</div>

{article.categories.includes("Uncategorized") && (

<select

defaultValue=""

onChange={(e) => handleCategoryChange(article.id, e.target.value)}

>

<option value="" disabled>Select category</option>

{Object.keys(categoryKeywords).map(cat => (

<option key={cat} value={cat}>{cat}</option>

))}

</select>

)}

</div>

))}

</div>

);

};

Step 5: Make sure your backend clearly handles category updates:

// Backend example using Express.js

app.post('/api/articles/:id/update-category', async (req, res) => {

const { id } = req.params;

const { category } = req.body;

await ArticlesCollection.updateOne(

{ id },

{ $set: { categories: [category] } }

);

res.json({ success: true });

});