Task Assignment

≈===============

================

This assignment focuses on enhancing a REST API demo project by adding new features and functionality. Below, I’ve outlined a plan and provided some code snippets to help you get started.

1. Display User Information for Each PostFirst, modify your JavaScript to fetch user information alongside posts.

You can use Promise.all to fetch posts and users simultaneously:async function fetchPostsAndUsers() {

Try {

Const [postsResponse, usersResponse] = await Promise.all([

Fetch(‘https://jsonplaceholder.typicode.com/posts’),

Fetch(‘https://jsonplaceholder.typicode.com/users’)

]);

Const posts = await postsResponse.json();

Const users = await usersResponse.json();

// Create a user lookup object

Const usersById = users.reduce((acc, user) => {

Acc[user.id] = user;

Return acc;

}, {});

displayPosts(posts, usersById);

} catch (error) {

Console.error(‘Error fetching posts and users:’, error);

displayError(‘Failed to load posts and users. Please try again later.’);

}

}

Function displayPosts(posts, usersById) {

Const postsContainer = document.getElementById(‘posts-container’);

postsContainer.innerHTML = ‘’; // Clear existing content

posts.forEach(post => {

const user = usersById[post.userId];

const postElement = document.createElement(‘div’);

postElement.className = ‘post’;

postElement.innerHTML = `

<h2>${post.title}</h2>

<p>${post.body}</p>

<p><strong>Author:</strong> ${user.name} (${user.email})</p>

`;

postElement.addEventListener(‘click’, () => fetchPostDetails(post.id));

postsContainer.appendChild(postElement);

});

}

2. Display Full Details of a Post Including CommentsAdd functionality to fetch and display the full details of a post when clicked, including its comments

:async function fetchPostDetails(postId) {

Try {

Const [postResponse, commentsResponse] = await Promise.all([

Fetch(`https://jsonplaceholder.typicode.com/posts/${postId}`),

Fetch(`https://jsonplaceholder.typicode.com/comments?postId=${postId}`)

]);

Const post = await postResponse.json();

Const comments = await commentsResponse.json();

displayPostDetails(post, comments);

} catch (error) {

Console.error(‘Error fetching post details:’, error);

displayError(‘Failed to load post details. Please try again later.’);

}

}

Function displayPostDetails(post, comments) {

Const postDetailsContainer = document.getElementById(‘post-details-container’);

postDetailsContainer.innerHTML = `

<h2>${post.title}</h2>

<p>${post.body}</p>

<h3>Comments</h3>

${comments.map(comment => `

<div class=”comment”>

<p><strong>${comment.name} (${comment.email}):</strong></p>

<p>${comment.body}</p>

</div>

`).join(‘’)}

`;

postDetailsContainer.style.display = ‘block’;

}

3. Improve Overall StylingEnhance the styling of your page using CSS. Add styles for the loading indicator, error message, and post details:/\* Add your styles here \*/

Body {

Font-family: Arial, sans-serif;

}

#posts-container {

Display: flex;

Flex-wrap: wrap;

Gap: 16px;

}

.post {

Border: 1px solid #ccc;

Padding: 16px;

Border-radius: 8px;

Width: 300px;

Cursor: pointer;

Transition: transform 0.3s;

}

.post:hover {

Transform: scale(1.05);

}

#post-details-container {

Display: none;

Border: 1px solid #ccc;

Padding: 16px;

Border-radius: 8px;

Margin-top: 20px;

}

.comment {

Border-top: 1px solid #eee;

Padding-top: 10px;

Margin-top: 10px;

}

.loading-indicator, .error-message {

Text-align: center;

Font-size: 1.2em;

Margin: 20px 0;

}

.error-message {

Color: red;

}HTML StructureEnsure your HTML structure accommodates the new features. Here’s an example:<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>

<title>REST API Demo Project</title>

<link rel=”stylesheet” href=”styles.css”>

</head>

<body>

<div id=”posts-container”></div>

<div id=”post-details-container”></div>

<div class=”loading-indicator” id=”loading-indicator”>Loading…</div>

<div class=”error-message” id=”error-message”></div>

<script src=”scripts.js”></script>

</body>

</html>

-------------------×-×-×-×-×-×-×-×-×-×-×-×-×-×----------------------