IMAGE RECOGNITION USING IBM CLOUD VISUAL RECOGNITION

PHASE5: PROJECT DOCUMENTATION AND SUBMISSION

Creating an image recognition system using a combination of technologies like Flash, HTML, CSS, and JavaScript can be an exciting project. Here’s a step-by-step guide on how you can achieve this, along with explanations and API integration:

Step 1: Set Up Your Development Environment:

HTML and CSS: Design a simple web interface with an upload button and a space to display recognized images.

JavaScript: Write code to handle user interactions and make API calls.

Flash: Utilize Flash to enhance the user experience, although please note that Flash technology has been largely deprecated, so it’s advisable to focus on HTML5 and JavaScript for modern web applications.

Step 2: Choose an Image Recognition API:

Select a suitable image recognition API that provides the functionality you need. Examples include Google Cloud Vision API, Amazon Rekognition, or IBM Watson Visual Recognition.

Step 3: HTML (index.html):

Html

<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

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

<title>Image Recognition</title>

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

</head>

<body>

<input type=”file” id=”imageUpload” accept=”image/\*”>

<div id=”result”></div>

<script src=”script.js”></script>

</body>

</html>

STEP4:CSS (styles.css):

Css

/\* Add your CSS styles here \*/

Body {

Font-family: Arial, sans-serif;

Display: flex;

Flex-direction: column;

Align-items: center;

Justify-content: center;

Height: 100vh;

}

#result {

Margin-top: 20px;

Padding: 10px;

Border: 1px solid #ccc;

Width: 80%;

Max-width: 600px;

White-space: pre-wrap;

}

STEP5:JavaScript (script.js):

Javascript

Document.getElementById(‘imageUpload’).addEventListener(‘change’, handleImageUpload);

Function handleImageUpload(event) {

Const file = event.target.files[0];

Const formData = new FormData();

formData.append(‘image’, file);

// Replace ‘API\_ENDPOINT\_URL’ and ‘YOUR\_API\_KEY’ with actual values

Fetch(‘API\_ENDPOINT\_URL’, {

Method: ‘POST’,

Body: formData,

Headers: {

‘Authorization’: ‘Bearer YOUR\_API\_KEY’,

}

})

.then(response => response.json())

.then(data => {

// Handle API response

displayRecognitionResult(data);

})

.catch(error => {

Console.error(‘Error:’, error);

Document.getElementById(‘result’).innerText = ‘Error occurred during recognition.’;

});

}

Function displayRecognitionResult(data) {

// Process and display API response, update the result div Document.getElementById(‘result’).innerText = JSON.stringify(data, null, 2);

}

Step 6: API Integration:

Replace ‘API\_ENDPOINT\_URL’ with the actual API endpoint provided by your chosen image recognition service.

Replace ‘YOUR\_API\_KEY’ with the authentication key/token required by the API service.

Step 7: Flash Integration:

Integrate your Flash components as needed. However, it’s essential to ensure compatibility with modern web standards and consider using HTML5 and JavaScript for interactive elements.

Remember to handle errors, provide user feedback, and style your interface for a better user experience. Testing and debugging are crucial throughout the development process.

Output:

When you open the HTML file in a browser, you’ll see a file input field.

Select an image file (such as a JPEG or PNG) using the file input.

The JavaScript code will send the image to the specified API endpoint for recognition.

The API response will be displayed below the file input field, showing the recognized content from the image.

Remember, the actual output will depend on the response from the image recognition API you’re using, and it will be displayed in the #result div as structured JSON data.

