**User Story: Feedback Form Integration for Crime App Using EmailJS**

**1. Title/Overview:**

**As a user**, I want a feedback form to report issues or provide suggestions so that I can communicate directly with the app developers and help improve my experience.

**2. Context and Problem Statement:**

* **Problem:** Users lacked a simple way to provide feedback on the app's functionality or content, and no efficient system existed to process such inputs.
* **Solution:** Implement a feedback form powered by **EmailJS** to handle feedback submissions seamlessly and efficiently via email.

**3. Features of the Feedback Form:**

* **Fields Included:**
  + Name (required)
  + Email (required, for follow-ups)
  + Message/Feedback content (required)
* **Validation:**
  + Ensures required fields are filled and email format is valid.
* **EmailJS Integration:**
  + Sends feedback data securely to a designated email address.
* **Acknowledgment Email:**
  + Sends an automatic confirmation email to the user, acknowledging their feedback.

**4. Workflow Demonstration:**

1. **User Interface:**
   * Simple and intuitive form with input fields and a "Submit" button.
2. **Backend Logic:**
   * On submission, form data is validated and sent using EmailJS services.
   * A confirmation email is triggered using EmailJS templates.
3. **Error Handling:**
   * Display clear error messages (e.g., invalid email format, network failure) and encourage users to retry.

**5. Technical Implementation:**

* **EmailJS Configuration:**
  + **Service ID:** Service ID from EmailJS account
  + **Template ID:** Template ID from EmailJS account
  + **Public Key**: Public Key from EmailJS account
* **Code Snippet Example:**

import React, { useState } from 'react';

import emailjs from 'emailjs-com';

import './ContactUsComponent.css';

const validateEmail = (email) => {

    // Simple regular expression to validate email format

    const emailPattern = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    return emailPattern.test(email);

};

const ContactUsComponent = () => {

    const [email, setEmail] = useState('');

    const [emailError, setEmailError] = useState('');

    const handleEmailBlur = () => {

        if (!validateEmail(email)) {

            setEmailError('Please enter a valid email address.');

        } else {

            setEmailError('');

        }

    };

    const sendEmail = (e) => {

        e.preventDefault(); // Prevent form refresh

        const form = e.target; // Reference to the form element

        const fromName = "Smart Communities Calgary Application team";

        const toName = form.name.value; // Get the value of the "name" input

        const replyTo = form.email.value; // Get the value of the "email" input

        const message = form.message.value; // Get the value of the "message" textarea

        // Validate the email on submit

        if (!validateEmail(replyTo)) {

            setEmailError('Please enter a valid email address.');

            return; // Prevent form submission if the email is invalid

        }

        const serviceID = ""; // Service Id from EmailJS

        const templateID = ""; // Template Id from EmailJS

        const publicKey =""; // Public Key from EmailJS

        // Sending email using emailjs.send

        emailjs

            .send(serviceID, templateID, {

                from\_name: fromName,

                to\_name: toName,

                reply\_to: replyTo,

                user\_message: message,

            }, publicKey)

            .then((result) => {

                console.log('Email sent successfully:', result.text);

                alert('Your message has been sent successfully!');

                form.reset(); // Optional: Reset the form after submission

            })

            .catch((error) => {

                console.error('Error sending email:', error.text);

                alert('There was an error sending your message. Please try again.');

            });

    };

    return (

        <form onSubmit={sendEmail}>

            <h2>Contact Us</h2>

            <input

                type="text"

                name="name"

                placeholder="Your Name"

                required

            />

            <input

                type="email"

                name="email"

                placeholder="Your Email"

                value={email}

                onBlur={handleEmailBlur} // Validate email when the user tabs out

                onChange={(e) => setEmail(e.target.value)} // Update email state

                required

            />

            {/\* Display email error if validation fails \*/}

            {emailError && <div className="error">{emailError}</div>}

            <textarea

                name="message"

                placeholder="Your Message"

                required

            />

            <button type="submit">Send</button>

        </form>

    );

};

export default ContactUsComponent;

**6. Impact and Outcome:**

* **Enhanced User Experience:**
  + Users now have a quick and direct way to provide feedback.
* **Streamlined Communication:**
  + Developers receive actionable feedback via email.
* **App Engagement:**
  + Active communication fosters trust and improves app satisfaction.

**7. Challenges and Lessons Learned:**

* Configuring and testing **EmailJS templates** for seamless integration.
* Ensuring proper error handling and security to prevent spam submissions.

**8. Future Improvements:**

* Add CAPTCHA for enhanced form security.
* Explore analytics or tracking mechanisms for submitted feedback.
* Allow users to upload screenshots or files for detailed feedback.