Here’s how you can easily demonstrate the **React-based contact form submission using Azure Functions**:

**Step-by-Step Demonstration**

**1. Create the Azure Function**

1. Open the Azure Portal.
2. Go to **Azure Functions** and click **Create Function App**.
3. Configure the function app:
   * Select a resource group.
   * Use a unique name for your function app (e.g., simple-contact-form).
   * Runtime stack: **Node.js**.
   * Hosting: Select **Consumption Plan**.
   * Click **Review + Create** and then **Create**.
4. Once the function app is created:
   * Go to the **Functions** section in the function app.
   * Click **+ Add Function**, select **HTTP Trigger**, and name it SubmitForm.

**2. Write the Function Code**

1. In the Azure Portal, edit the index.js file of the function:

module.exports = async function (context, req) {

context.log('JavaScript HTTP trigger function processed a request.');

// Extract the name and email from query or request body

const name = (req.query.name || (req.body && req.body.name));

const email = (req.query.email || (req.body && req.body.email));

// Check if both name and email are provided for the personalized response

let responseMessage;

if (name && email) {

responseMessage = `Hello, ${name}, your email address is ${email}. This HTTP triggered function executed successfully.`;

} else {

responseMessage = "This HTTP triggered function executed successfully. Pass both name and email in the query string or in the request body for a personalized response.";

}

// Set the response with the message

context.res = {

// status: 200, /\* Defaults to 200 \*/

body: responseMessage

};

};

1. Save the file.
2. Test the get request by adding query parameters :

Name: rakesh

Email testrakesh

1. Test the function using the **Test/Run** button in the portal for post request:
   * Choose POST method.
   * Use a JSON payload:

{

"name": "John Doe",

"email": "john@example.com",

"message": "Hello!"

}

* + Check the output. You should see the success response.

**3. React Frontend**

1. **Create a React App**:

npx create-react-app simple-contact-form

cd simple-contact-form

npm start

1. **Add a Contact Form** in src/App.js: Replace the content of App.js with:

import React, { useState } from 'react';

function App() {

const [formData, setFormData] = useState({ name: '', email: '', message: '' });

const [response, setResponse] = useState('');

const handleChange = (e) => {

setFormData({ ...formData, [e.target.name]: e.target.value });

};

const handleSubmit = async (e) => {

e.preventDefault();

const res = await fetch('<Azure Function URL>', {

method: 'POST',

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

body: JSON.stringify(formData),

});

const data = await res.text();

setResponse(data);

};

return (

<div>

<h1>Contact Form</h1>

<form onSubmit={handleSubmit}>

<input name="name" placeholder="Name" onChange={handleChange} />

<input name="email" placeholder="Email" onChange={handleChange} />

<textarea name="message" placeholder="Message" onChange={handleChange}></textarea>

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

</form>

{response && <p>{response}</p>}

</div>

);

}

export default App;

1. Replace <Azure Function URL> with the URL from your Azure Function (you’ll find this in the Azure Portal under **Function URL**).
2. Start the app:

bash

Copy code

npm start

1. we need to allow other origins as well , so enable CORS setting on azure function and add an origin to allowed origins in the CORS :

http://localhost:3000

**4. Test the Demo**

1. Open the React app in your browser.
2. Fill in the form and click **Submit**.
3. Observe the response below the form, showing the message from the Azure Function.

**Key Talking Points During the Demo**

1. **What is Happening?**
   * React sends a POST request to the Azure Function with form data.
   * Azure Function validates the data and sends a response back.
2. **Why Azure Functions?**
   * Serverless: No need to manage infrastructure.
   * Quick and scalable backend for simple applications.
3. **Scalability**:
   * Azure Functions can handle multiple requests simultaneously and scale as needed.