To register and process the messages from your contact form, you'll need to implement a backend system. Here's a general approach you can take:

1. Set up a server-side script: Create a server-side script (e.g., in PHP, Python, Node.js) to handle form submissions.
2. Update the form's action attribute: Change the form's action to point to your server-side script:

html

Copy

<form action="/submit\_form.php" method="POST" class="row g-3 p-lg-5 p-4 bg-white theme-shadow">

1. Implement server-side processing: In your server-side script (e.g., submit\_form.php), process the form data:

php

Copy

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$firstName = $\_POST['firstName'];

$lastName = $\_POST['lastName'];

$email = $\_POST['email'];

$subject = $\_POST['subject'];

$message = $\_POST['message'];

*// Store in database or send email*

*// ...*

*// Redirect back to the form with a success message*

header("Location: index.html?status=success");

}

?>

1. Store the data: You can store the form data in a database or send it via email.
2. Add name attributes to your form inputs: Ensure each input has a name attribute:

html

Copy

<input type="text" name="firstName" class="form-control" placeholder="Enter first name">

1. Implement client-side validation: Use JavaScript to validate the form before submission:

javascript

Copy

document.querySelector('form').addEventListener('submit', function(e) {

*// Perform validation*

*// If validation fails, call e.preventDefault() to stop form submission*

});

1. Handle form submission with AJAX (optional): For a smoother user experience, you can submit the form using AJAX:

javascript

Copy

document.querySelector('form').addEventListener('submit', function(e) {

e.preventDefault();

fetch('/submit\_form.php', {

method: 'POST',

body: new FormData(this)

})

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

.then(data => {

*// Handle response*

});

});

1. Implement security measures:
   * Use CSRF tokens to prevent cross-site request forgery.
   * Implement rate limiting to prevent spam.
   * Sanitize and validate all input data on the server side.

Remember to comply with data protection regulations (like GDPR) when collecting and storing personal information. Also, consider using a CAPTCHA system to prevent bot submissions.

This is a basic outline. The exact implementation will depend on your server setup, chosen backend technology, and specific requirements.