Got it! You want a full-stack solution where:

1. A frontend HTML page collects the email.

2. A backend server processes the request and triggers API Gateway.

3. API Gateway invokes the Lambda function to store the email in DynamoDB.

Solution Overview:

- Frontend: HTML + JavaScript (Fetch API)

- Backend: Express.js (Node.js) or Python Flask

- Database: DynamoDB

- AWS Services Used: Lambda, API Gateway, DynamoDB

---

1. API Gateway Setup:

Create an HTTP API in API Gateway that:

- Accepts POST requests to /submit-email.

- Integrates with the Lambda function.

- Has CORS enabled to allow frontend access.

---

2. Lambda Function (Python):

This function will receive the email and store it in DynamoDB.

Lambda Function Code (Python)

```
import json
import boto3
import re

dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table("EmailsTable")

def validate_email(email):
    pattern = r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"
    return re.match(pattern, email)

def lambda_handler(event, context):
    try:
        body = json.loads(event["body"])
        email = body.get("email")

        if not email or not validate_email(email):
            return {"statusCode": 400, "body": json.dumps({"message": "Invalid email format"})}
```

```
        table.put_item(Item={"email": email})

        return {"statusCode": 200, "body": json.dumps({"message": "Email stored successfully"})}

    except Exception as e:
            return {"statusCode": 500, "body": json.dumps({"message": "Internal server error", "error": str(e)})}
```

---

3. Backend Server (Node.js with Express):

Your backend will expose an API that the frontend can call. It will forward requests to API Gateway.

Install Express.js:

```
npm init -y
npm install express axios cors dotenv
```

Create server.js:

```
require('dotenv').config();
const express = require("express");
const axios = require("axios");
const cors = require("cors");
```

```
const app = express();
app.use(express.json());
app.use(cors());

const API_GATEWAY_URL = process.env.API_GATEWAY_URL;

app.post("/submit-email", async (req, res) => {
    try {
        const { email } = req.body;

        if (!email) {
            return res.status(400).json({ message: "Email is required" });
        }

        const response = await axios.post(API_GATEWAY_URL, { email });

        res.json(response.data);
    } catch (error) {
        res.status(500).json({ message: "Server error", error: error.message });
    }
});

app.listen(3000, () => console.log("Server running on port 3000"));
```

---

4. Frontend (HTML + JavaScript):

Create a simple HTML page to collect the email and send it to your backend.

Create index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Email Submission</title>
</head>
<body>
    <h2>Subscribe with your Email</h2>
    <form id="emailForm">
        <input type="email" id="emailInput" placeholder="Enter your email" required>
        <button type="submit">Submit</button>
    </form>
    <p id="message"></p>

    <script>
        document.getElementById("emailForm").addEventListener("submit", async function(event) {
            event.preventDefault();
            const email = document.getElementById("emailInput").value;
```

```
        const response = await fetch("http://localhost:3000/submit-email", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ email })
        });

        const result = await response.json();
        document.getElementById("message").textContent = result.message;
    });
  </script>
</body>
</html>
```

---

5. Running the Project:

Backend:

---

1. Create a .env file and add your API Gateway URL:

API_GATEWAY_URL=https://your-api-id.execute-api.region.amazonaws.com/submit-email

2. Run the backend:

node server.js

Frontend:

Just open index.html in a browser.

Final Flow:

---

1. User submits email -> JavaScript sends it to the backend (Express.js).
2. Backend forwards the email to API Gateway.
3. API Gateway triggers the Lambda function.
4. Lambda validates & stores the email in DynamoDB.
5. Success message is displayed.

Would you like to deploy the backend to AWS as well?