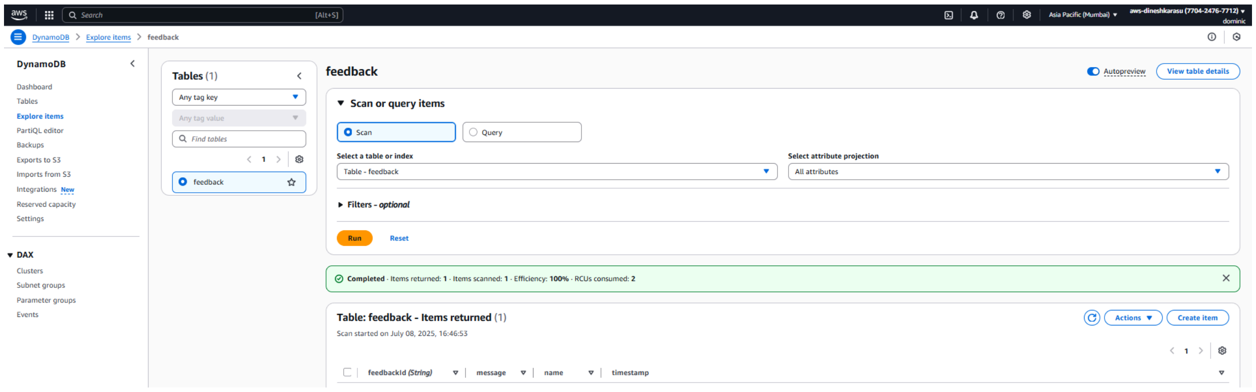
Use case	Use case Description
Serverless Feedback Application	Frontend form (HTML/JS) -> API Gateway -> Lambda -> DynamoDB.
	Store and retrieve feedback using AWS services

Approach :

1. Create a DynamoDB Table named feedback
2. Create a Lambda Function Add IAM permissions for the function to get item and put item
3. Implement Logic in lambda to store and retrieve feedbacks
4. Create an API Gateway to trigger Lambda for storing and retrieving
5. Build a Simple Frontend (HTML + JS)
6. Create a form to collect feedback (e.g., name, message)
7. Dump the files in an S3 bucket and enable static website hosting
8. Test



feedback

Scan or query items

Scan [✓] Query

Select a table or index: Table - feedback

Select attribute projection: All attributes

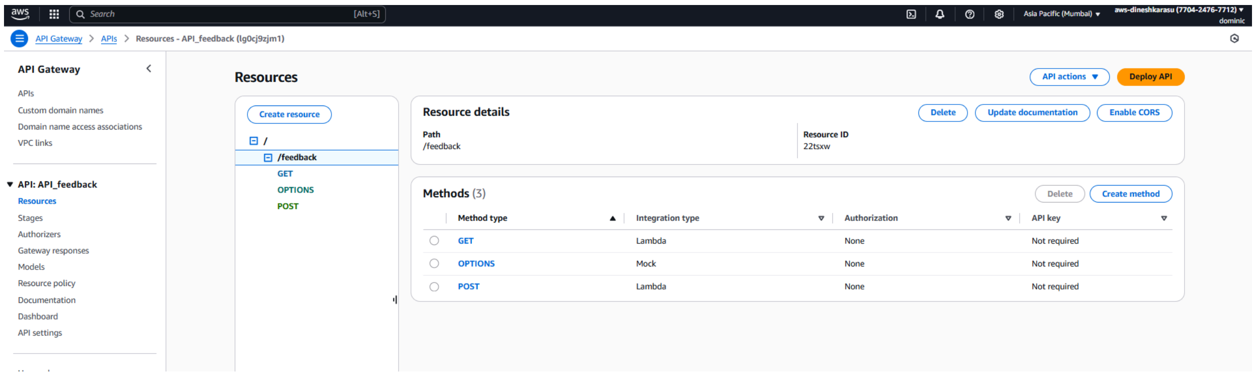
Filters - optional

Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2

Table: feedback - Items returned (1)

Scan started on July 08, 2025, 16:46:53

feedbackId (String)	message	name	timestamp



Resources

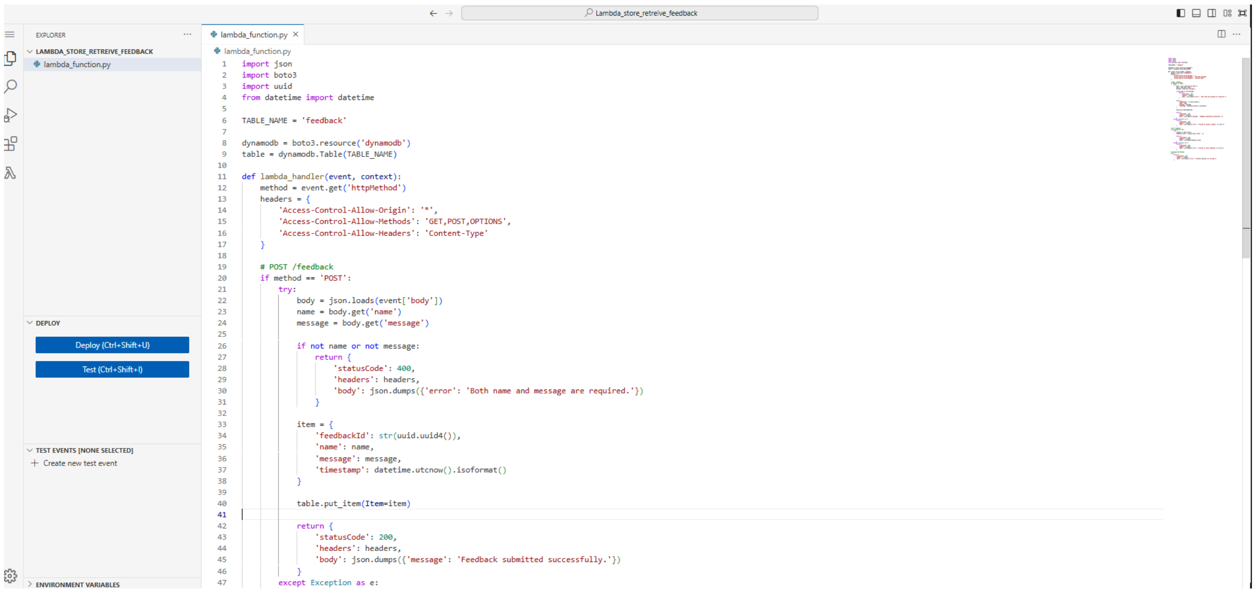
Resource details

Path: /feedback

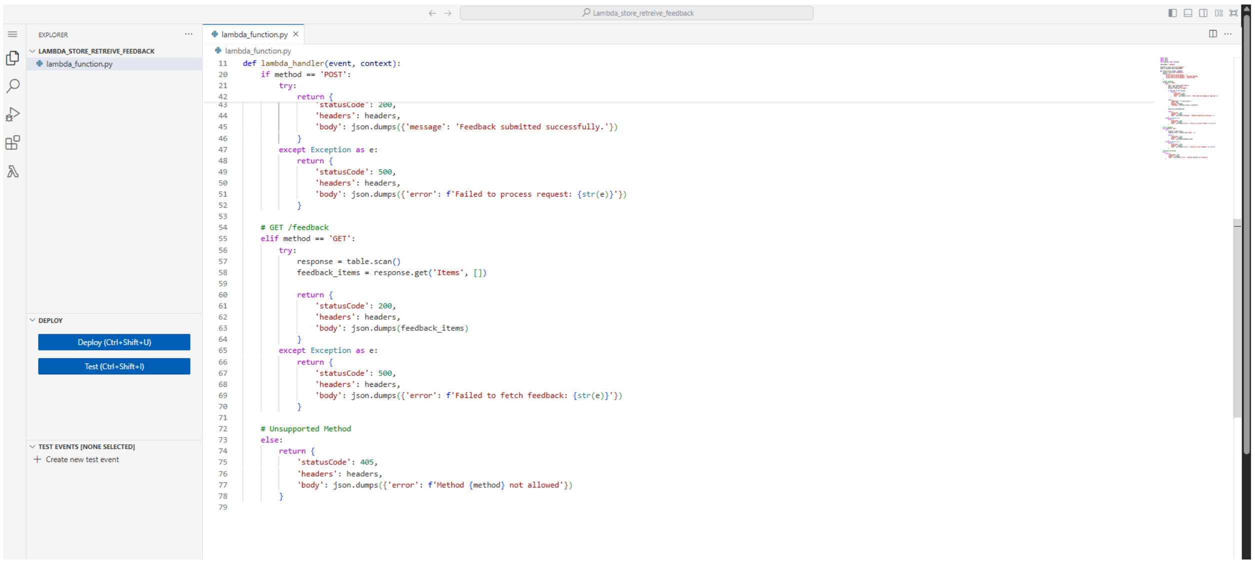
Resource ID: 22tsxw

Methods (3)

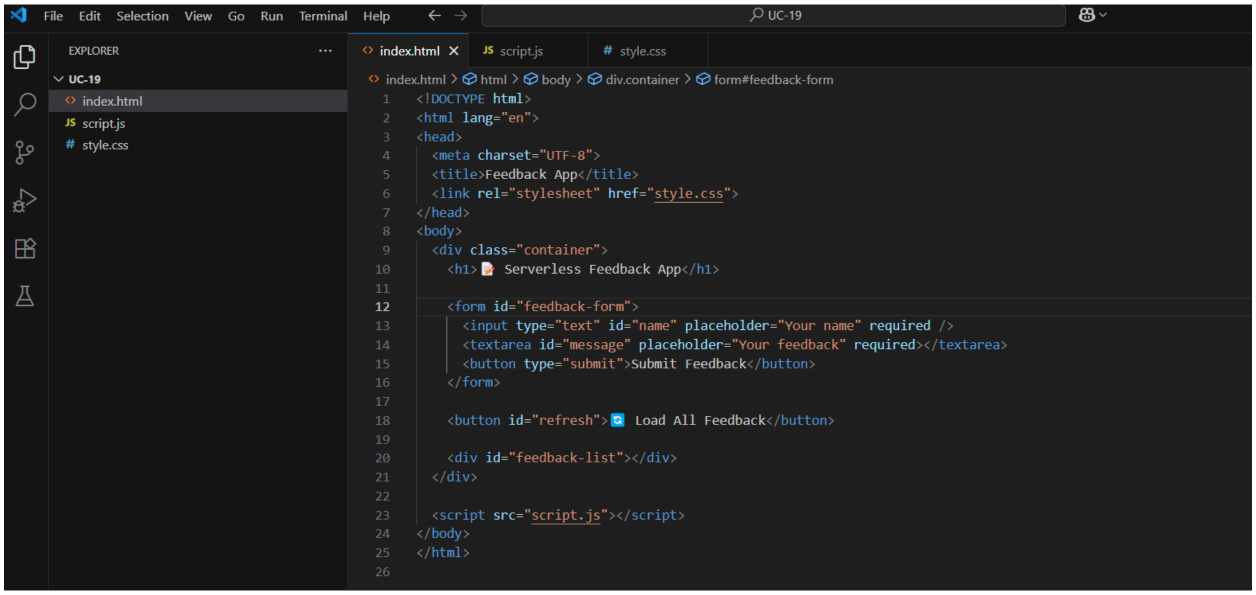
Method type	Integration type	Authorization	API key
GET	Lambda	None	Not required
OPTIONS	Mock	None	Not required
POST	Lambda	None	Not required



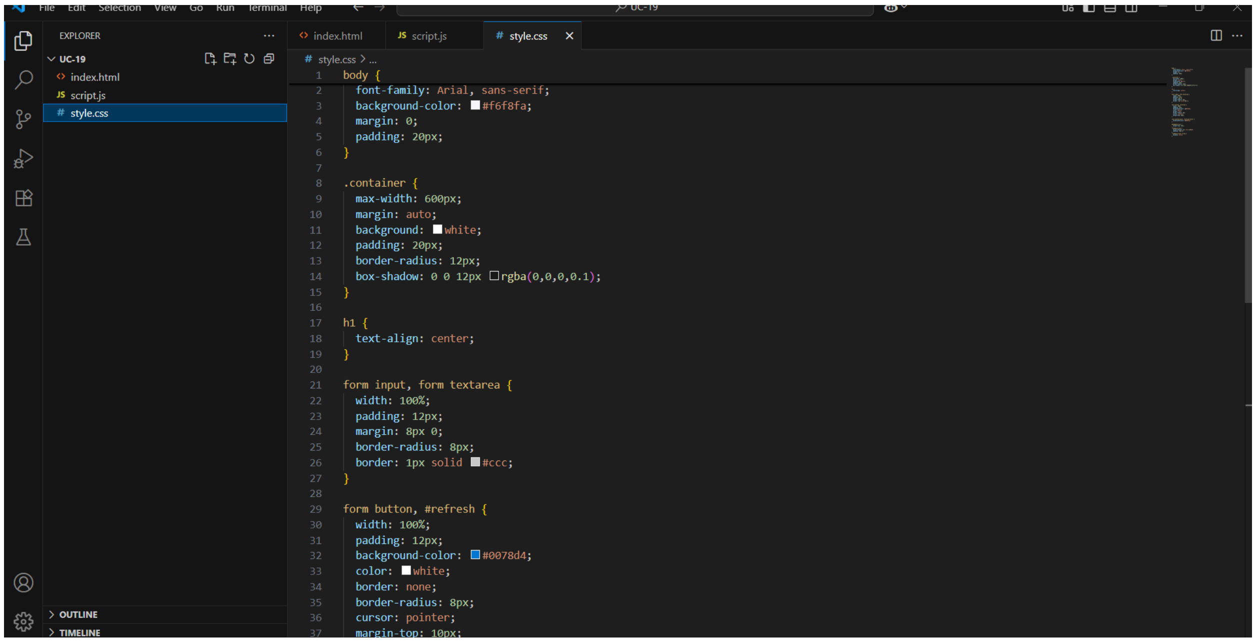
```python
import json
import boto3
import uuid
from datetime import datetime

TABLE_NAME = 'feedback'

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table(TABLE_NAME)

def lambda_handler(event, context):
    method = event.get('httpMethod')
    headers = {
        'Access-Control-Allow-Origin': '*',
        'Access-Control-Allow-Methods': 'GET,POST,OPTIONS',
        'Access-Control-Allow-Headers': 'Content-Type'
    }

    # POST /feedback
    if method == 'POST':
        try:
            body = json.loads(event['body'])
            name = body.get('name')
            message = body.get('message')

            if not name or not message:
                return {
                    'statusCode': 400,
                    'headers': headers,
                    'body': json.dumps({'error': 'Both name and message are required.'})
                }

            item = {
                'feedbackId': str(uuid.uuid4()),
                'name': name,
                'message': message,
                'timestamp': datetime.utcnow().isoformat()
            }

            table.put_item(Item=item)

            return {
                'statusCode': 200,
                'headers': headers,
                'body': json.dumps({'message': 'Feedback submitted successfully.'})
            }
        except Exception as e:
```



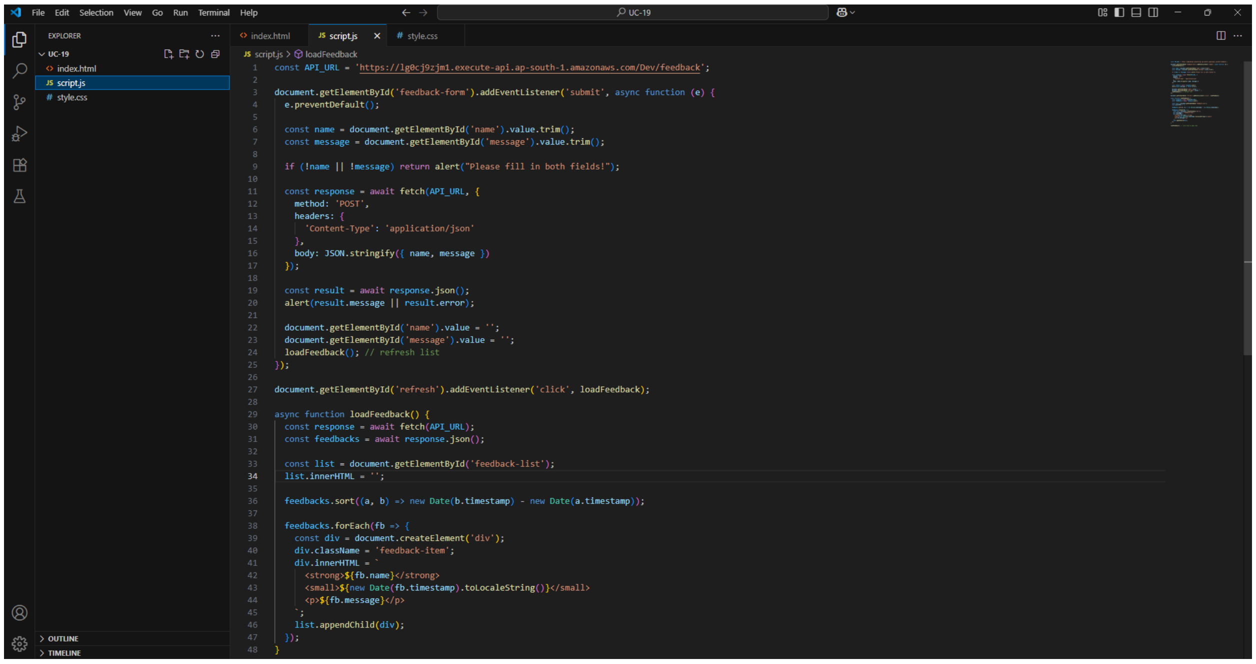
```python
def lambda_handler(event, context):
    if method == 'POST':
        try:
            return {
                'statusCode': 200,
                'headers': headers,
                'body': json.dumps({'message': 'Feedback submitted successfully.'})
            }
        except Exception as e:
            return {
                'statusCode': 500,
                'headers': headers,
                'body': json.dumps({'error': f'Failed to process request: {str(e)}'})
            }

    # GET /feedback
    elif method == 'GET':
        try:
            response = table.scan()
            feedback_items = response.get('Items', [])

            return {
                'statusCode': 200,
                'headers': headers,
                'body': json.dumps(feedback_items)
            }
        except Exception as e:
            return {
                'statusCode': 500,
                'headers': headers,
                'body': json.dumps({'error': f'Failed to fetch feedback: {str(e)}'})
            }

    # Unsupported Method
    else:
        return {
            'statusCode': 405,
            'headers': headers,
            'body': json.dumps({'error': f'Method {method} not allowed'})
        }

```



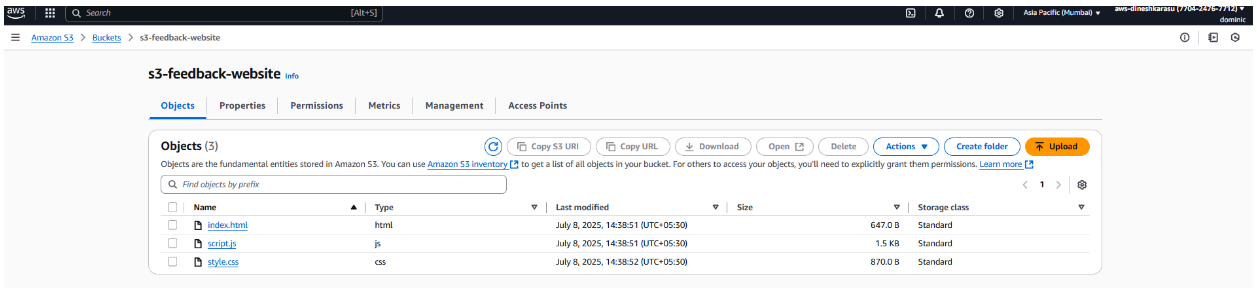
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Feedback App</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>📝 Serverless Feedback App</h1>

    <form id="feedback-form">
      <input type="text" id="name" placeholder="Your name" required />
      <textarea id="message" placeholder="Your feedback" required></textarea>
      <button type="submit">Submit Feedback</button>
    </form>

    <button id="refresh">🔄 Load All Feedback</button>

    <div id="feedback-list"></div>
  </div>

  <script src="script.js"></script>
</body>
</html>

```



```css
body {
  font-family: Arial, sans-serif;
  background-color: #f6f8fa;
  margin: 0;
  padding: 20px;
}

.container {
  max-width: 600px;
  margin: auto;
  background: white;
  padding: 20px;
  border-radius: 12px;
  box-shadow: 0 0 12px rgba(0,0,0,0.1);
}

h1 {
  text-align: center;
}

form input, form textarea {
  width: 100%;
  padding: 12px;
  margin: 8px 0;
  border-radius: 8px;
  border: 1px solid #ccc;
}

form button, #refresh {
  width: 100%;
  padding: 12px;
  background-color: #0078d4;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  margin-top: 10px;
```



```javascript
const API_URL = 'https://lg0cj9zjm1.execute-api.ap-south-1.amazonaws.com/Dev/feedback';

document.getElementById('feedback-form').addEventListener('submit', async function (e) {
  e.preventDefault();

  const name = document.getElementById('name').value.trim();
  const message = document.getElementById('message').value.trim();

  if (!name || !message) return alert("Please fill in both fields!");

  const response = await fetch(API_URL, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ name, message })
  });

  const result = await response.json();
  alert(result.message || result.error);

  document.getElementById('name').value = '';
  document.getElementById('message').value = '';
  loadFeedback(); // refresh list
});

document.getElementById('refresh').addEventListener('click', loadFeedback);

async function loadFeedback() {
  const response = await fetch(API_URL);
  const feedbacks = await response.json();

  const list = document.getElementById('feedback-list');
  list.innerHTML = '';

  feedbacks.sort((a, b) => new Date(b.timestamp) - new Date(a.timestamp));

  feedbacks.forEach(fb => {
    const div = document.createElement('div');
    div.className = 'feedback-item';
    div.innerHTML = `
      <strong>${fb.name}</strong>
      <small>${new Date(fb.timestamp).toLocaleString()}</small>
      <p>${fb.message}</p>
    `;
    list.appendChild(div);
  });
}
```

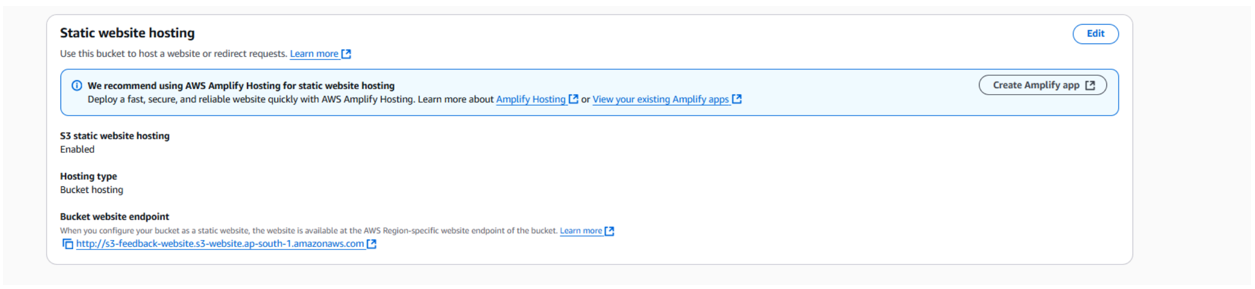


s3-feedback-website

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
index.html	html	July 8, 2025, 14:38:51 (UTC+05:30)	647.0 B	Standard
script.js	js	July 8, 2025, 14:38:51 (UTC+05:30)	1.5 KB	Standard
style.css	css	July 8, 2025, 14:38:52 (UTC+05:30)	870.0 B	Standard



Static website hosting

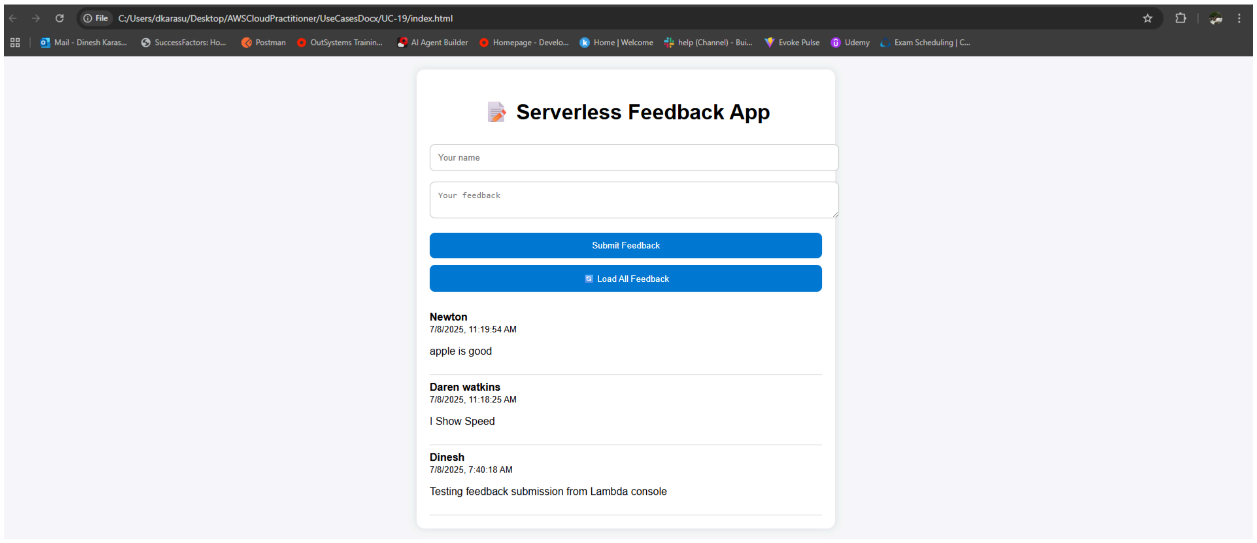
Use this bucket to host a website or redirect requests. Learn more

We recommend using AWS Amplify Hosting for static website hosting
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about Amplify Hosting or View your existing Amplify apps

S3 static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://s3-feedback-website.s3-website.ap-south-1.amazonaws.com



Serverless Feedback App

Newton
7/8/2025, 11:19:54 AM

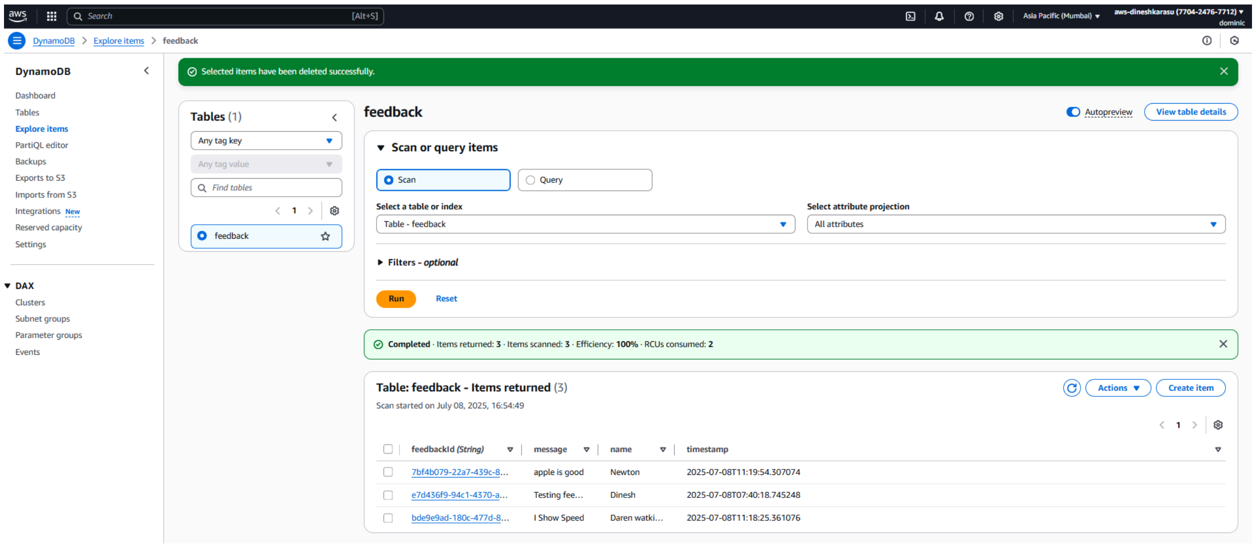
apple is good

Daren watkins
7/8/2025, 11:18:25 AM

I Show Speed

Dinesh
7/8/2025, 7:40:18 AM

Testing feedback submission from Lambda console



Selected items have been deleted successfully.

feedback

Scan or query items

[✓ Scan] Query

Select a table or index: Table - feedback

Select attribute projection: All attributes

Filters - *optional*

Completed · Items returned: **3** · Items scanned: **3** · Efficiency: **100%** · RCUs consumed: **2**

Table: feedback - Items returned (3)

Scan started on July 08, 2025, 16:54:49

feedbackId *(String)*	message	name	timestamp
7bf4b079-22a7-439c-8...	apple is good	Newton	2025-07-08T11:19:54.307074
e7d436f9-94c1-4370-a...	Testing fee...	Dinesh	2025-07-08T07:40:18.745248
bde9e9ad-180c-477d-8...	I Show Speed	Daren watki...	2025-07-08T11:18:25.361076