Serverless Feedback Collection System using AWS

Aim:

To build a simple serverless feedback collection system using core AWS services like S3, API Gateway, Lambda, and DynamoDB that can accept user feedback through a web form and store it securely in a database without requiring any backend server.

Problem Statement:

Collecting user feedback usually requires hosting a backend server, setting up a database, and managing infrastructure. This increases cost and complexity. A serverless approach simplifies deployment by using AWS managed services, offering a reliable and low-maintenance alternative for real-time feedback collection.

Key Benefits:

- **Serverless Architecture** — No servers to maintain or configure.
- **Cost-Effective** — Pay only for usage (Lambda invocations, API calls).
- **Easy to Deploy** — Minimal configuration with AWS Console.
- **Real-time Storage** — Feedback saved instantly in DynamoDB.
- **Browser-Friendly** — Form hosted directly via S3 as a static site.
- **Built-in Monitoring** — CloudWatch logs used for debugging.

AWS Services Used:

Core Services:

- **Amazon S3** – Hosts the static HTML feedback form.
- **Amazon API Gateway** – Provides the HTTP endpoint for form submission.
- **AWS Lambda** – Processes form data and stores it into DynamoDB.
- **Amazon DynamoDB** – Stores feedback entries.

Supporting Services:

- **AWS IAM** – Manages permissions between Lambda and DynamoDB.
- **Amazon CloudWatch** – Logs Lambda events for debugging.

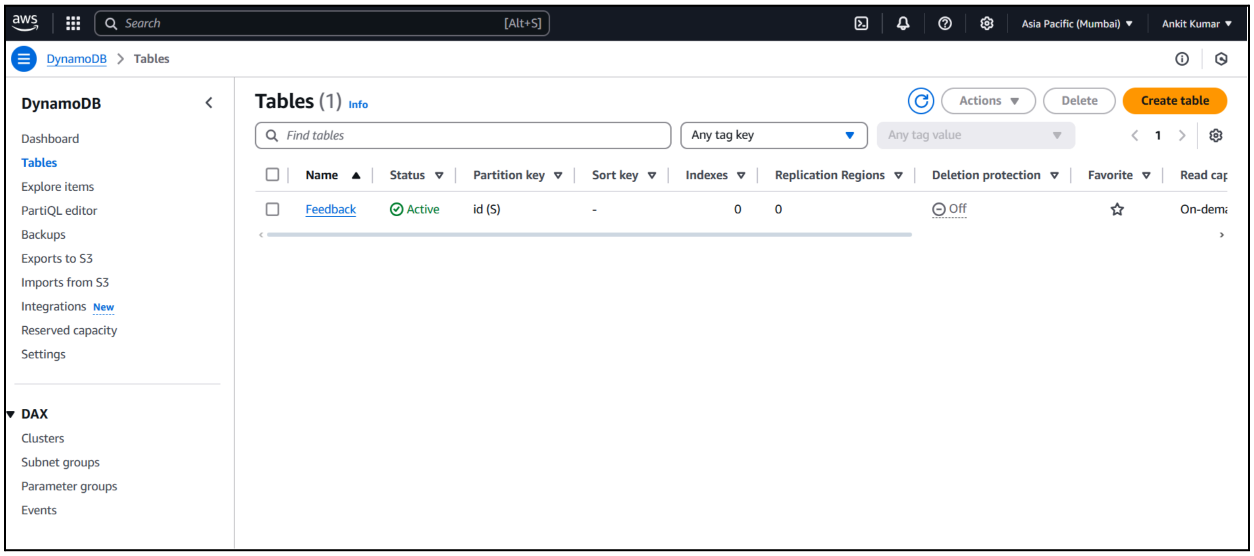
Project Workflow:

1. Users access the feedback form via an S3-hosted webpage.
2. On form submission, JavaScript sends a POST request to API Gateway.
3. API Gateway triggers a Lambda function.
4. Lambda parses the feedback and inserts it into a DynamoDB table.

Implementation Steps:

1. Create DynamoDB Table

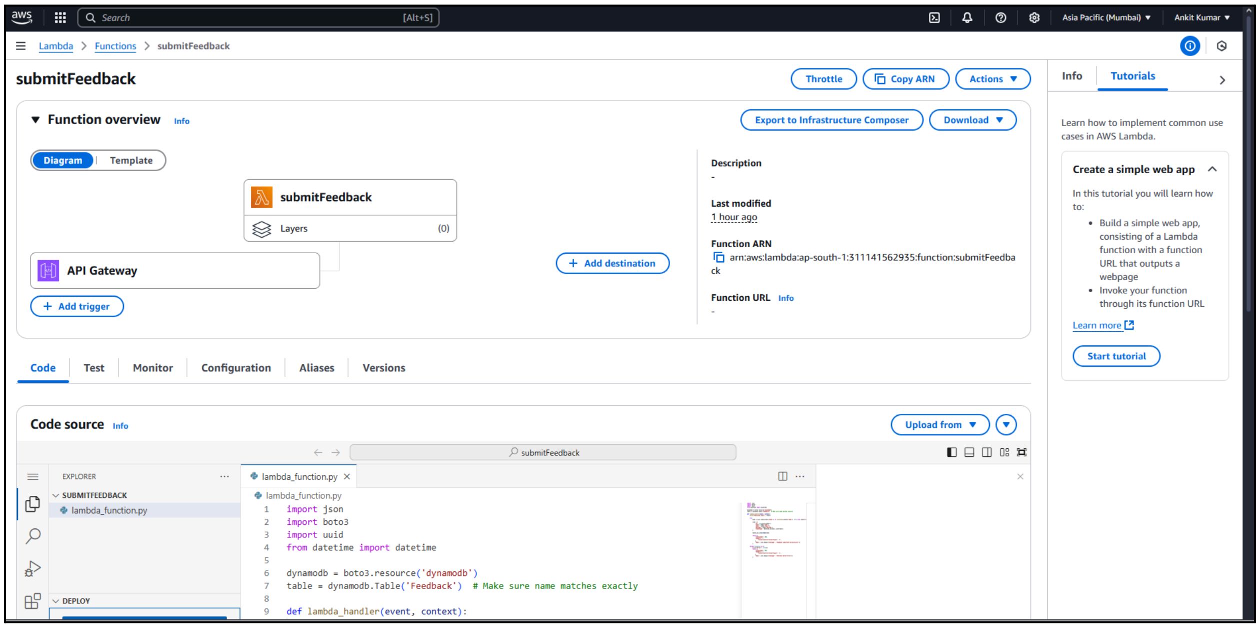
- **Table Name:** Feedback
- **Primary Key:** id (String)



	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read cap
	Feedback	Active	id (S)	-	0	0	Off		On-dem

2. Lambda Function

- Parses request body (JSON)
- Generates id and timestamp
- Inserts item into DynamoDB



submitFeedback

Description
-

Last modified
1 hour ago

Function ARN
arn:aws:lambda:ap-south-1:311141562935:function:submitFeedback

Function URL
-

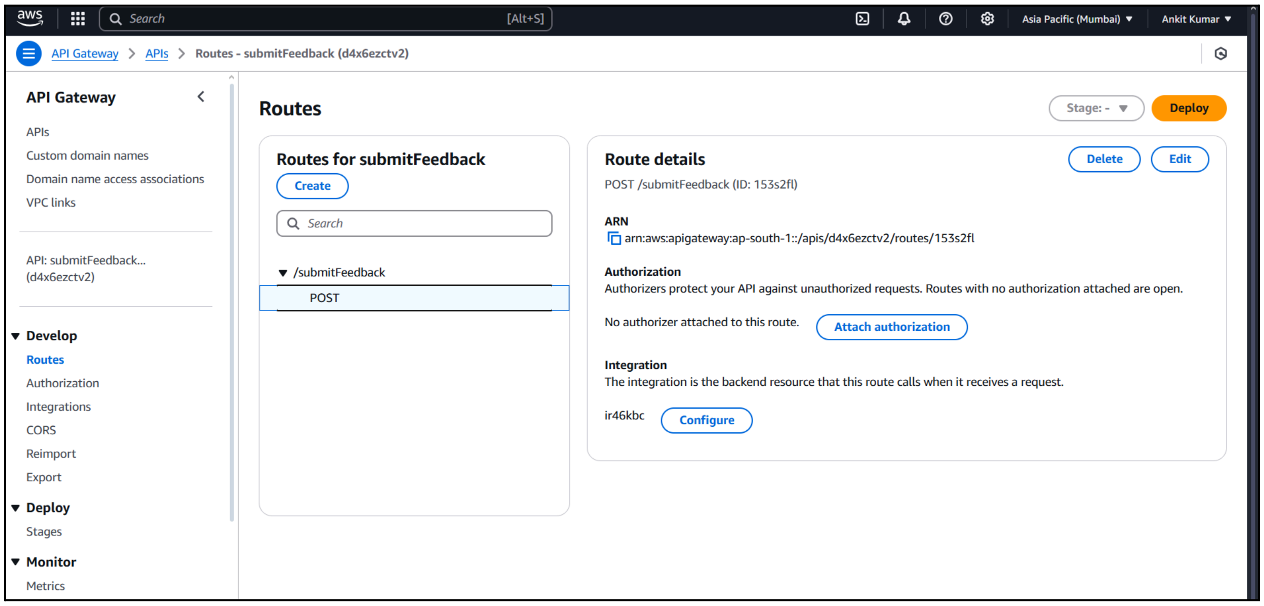
```
import json
import boto3
import uuid
from datetime import datetime

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Feedback')  # Make sure name matches exactly

def lambda_handler(event, context):
```

3. Configure API Gateway

- **Route:** POST /submitFeedback
- **Integration:** Lambda function
- **CORS Enabled** for browser access



Route details

POST /submitFeedback (ID: 153s2fl)

ARN

arn:aws:apigateway:ap-south-1::/apis/d4x6ezctv2/routes/153s2fl

Authorization

Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open.

No authorizer attached to this route.

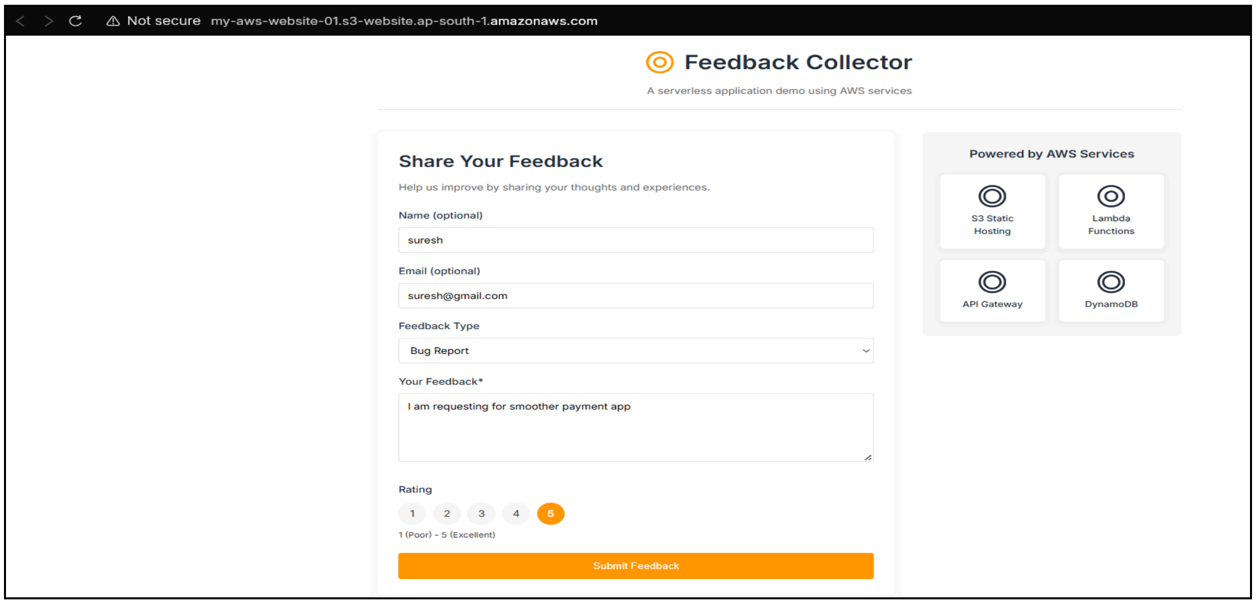
Integration

The integration is the backend resource that this route calls when it receives a request.

ir46kbc

4. Uploaded index.html to S3

- Static website hosting enabled
- Web Page uses fetch() to submit data to the API Gateway endpoint


Not secure my-aws-website-01.s3-website.ap-south-1.amazonaws.com
Feedback Collector
A serverless application demo using AWS services
Powered by AWS Services
S3 Static Hosting
Lambda Functions
API Gateway
DynamoDB
Submit Feedback

Share Your Feedback

Help us improve by sharing your thoughts and experiences.

Name (optional): suresh

Email (optional): suresh@gmail.com

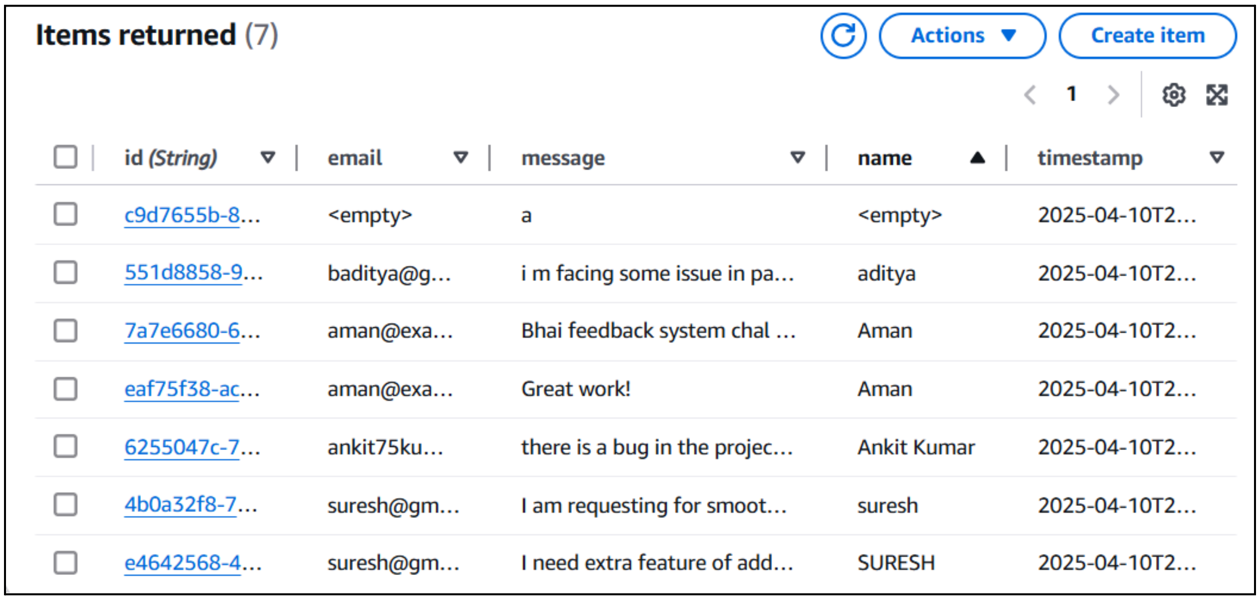
Feedback Type: Bug Report

Your Feedback*: I am requesting for smoother payment app

Rating: 1 2 3 4 [✓ 5]

1 (Poor) - 5 (Excellent)

5. Data entered via our portal of feedback mechanism.



Items returned (7)

id (String)	email	message	name	timestamp
c9d7655b-8…	<empty>	a	<empty>	2025-04-10T2…
551d8858-9…	baditya@g…	i m facing some issue in pa…	aditya	2025-04-10T2…
7a7e6680-6…	aman@exa…	Bhai feedback system chal …	Aman	2025-04-10T2…
eaf75f38-ac…	aman@exa…	Great work!	Aman	2025-04-10T2…
6255047c-7…	ankit75ku…	there is a bug in the projec…	Ankit Kumar	2025-04-10T2…
4b0a32f8-7…	suresh@gm…	I am requesting for smoot…	suresh	2025-04-10T2…
e4642568-4…	suresh@gm…	I need extra feature of add…	SURESH	2025-04-10T2…

Conclusion:

This project demonstrates a lightweight serverless solution for real-time feedback collection. It uses a combination of AWS services with minimal configuration and no server hosting, making it ideal for academic demos, simple internal tools, or prototype applications.

Use Cases:

- Mini projects & college demos
- Event or course feedback collection
- Internal suggestion boxes
- AWS learning projects