

# **Neocracy Travel Connect**

**(Cloud full-stack, serverless contact and  
booking web application)**



**Project By: Harjot Singh**  
**Trainer: Saurabh Dwivedi**

# Project Report: Neocracy Travel Connect

## Project Overview

**Neocracy Travel Connect** is a full-stack, serverless contact and booking web application designed for **Neocracy India Travel**, a travel service provider. The project allows users to explore destinations and directly submit travel queries through a responsive contact form. The entire backend is built using **AWS** serverless technologies ensuring scalability, zero server management, and cost-efficiency.

---

## Key Features

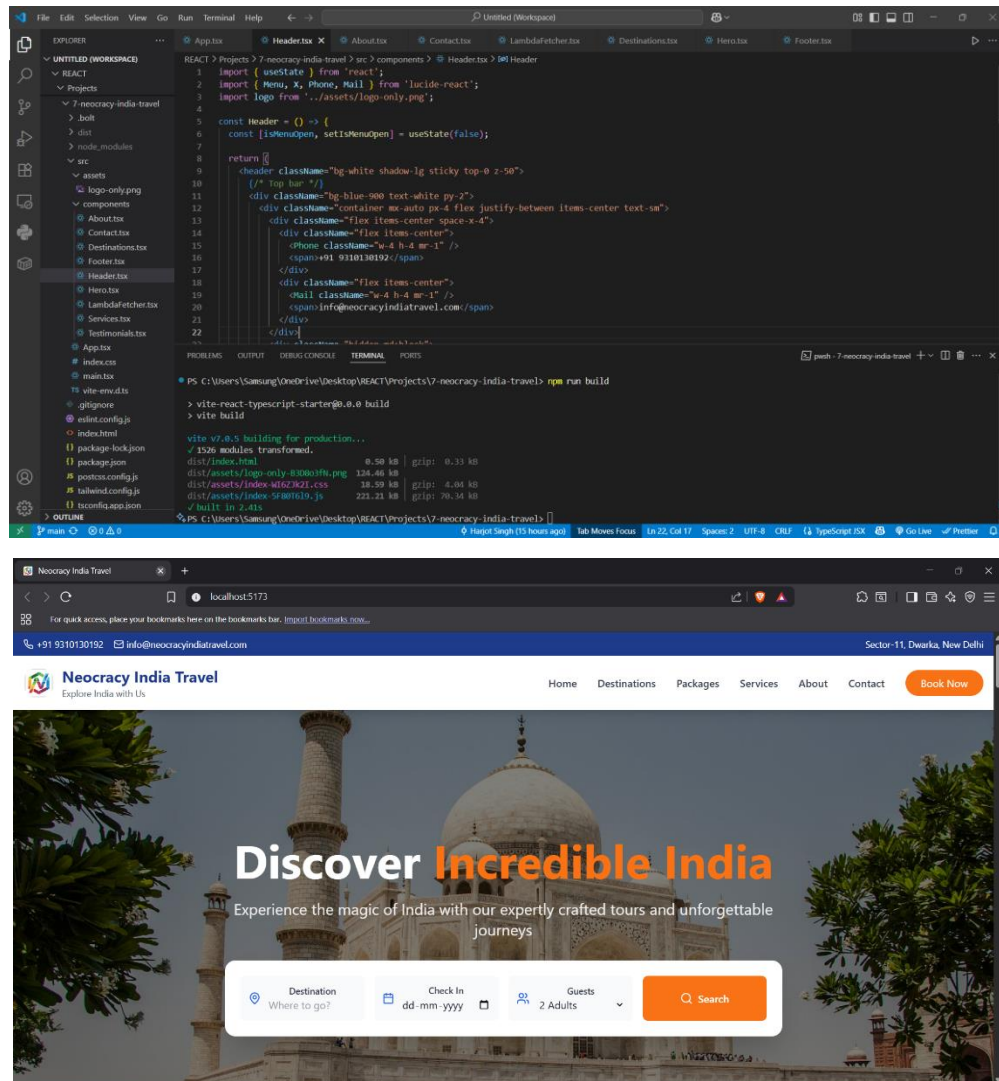
- Responsive front-end travel website built with **React + TypeScript + Tailwind CSS**
  - Static site hosted on **AWS S3** with a live custom endpoint
  - Functional contact form with the following capabilities:
    - Collects user inputs (name, email, phone, destination, message)
    - Sends form data to **AWS Lambda** via **API Gateway**
    - **Email notification** sent through **Amazon SNS**
    - Data saved into **Amazon DynamoDB**
  - Backend entirely serverless (no EC2 or manual servers involved)
  - Secured access via **IAM Roles**
- 

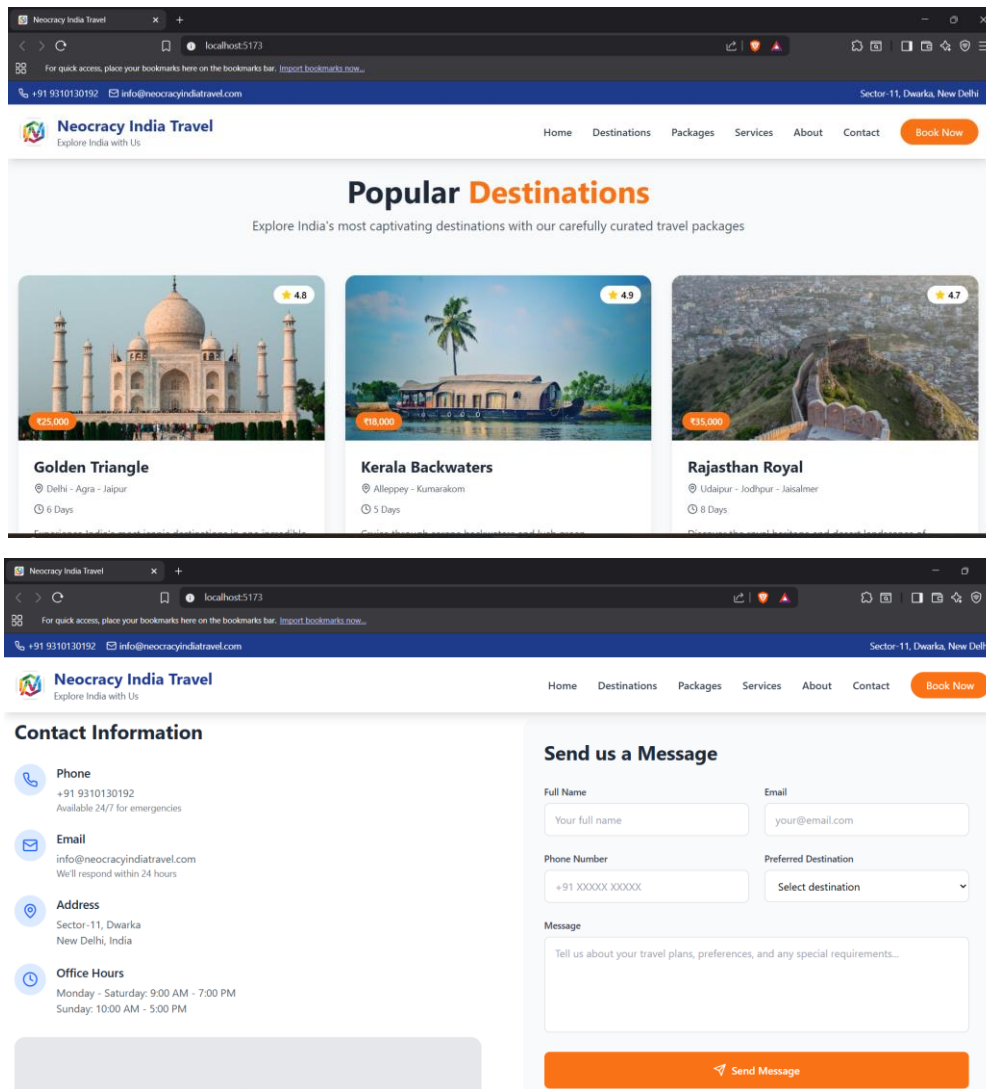
## Technologies Used

- **Frontend:** React, TypeScript, Tailwind CSS
  - **Hosting:** AWS S3 (Static Website Hosting)
  - **Backend:** AWS Lambda (Python 3)
  - **API Gateway:** For routing frontend requests to Lambda
  - **SNS:** For sending real-time email alerts
  - **DynamoDB:** For persisting form submissions
  - **IAM Roles:** For secure resource permissions
  - **Axios:** For HTTP requests from frontend to backend
-

## Implementation Steps:

### 1. Created a React project using Vite with TypeScript support





## 2. Used Axios to handle HTTP POST requests from React to API Gateway

```

File Edit Selection View Go Run Terminal Help
Unsaved (Workspace)

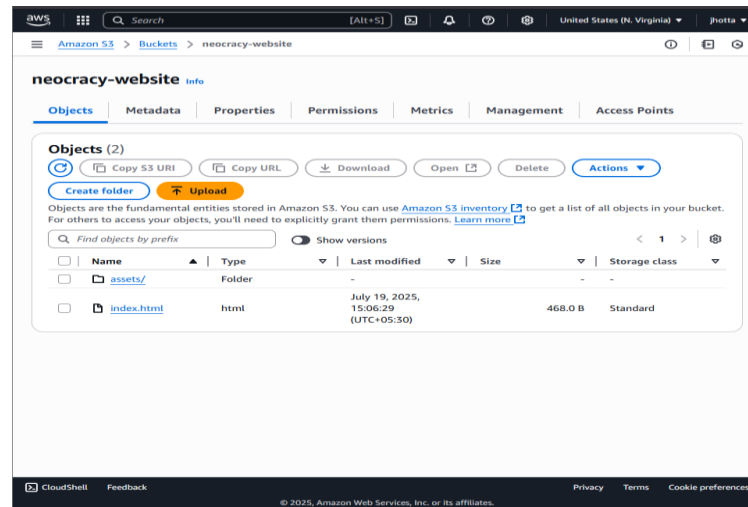
EXPLORER
  REACT
    Projects
      2 - todo-app-version-one
      3 - calculator-version-one
      4 - calculator-version-two
      5 - todo-app-version-three
      6 - neocracy-india-travel
        bolt
        dist
        node_modules
        src
        assets
        logo-only.png
        components
          About.jsx
          Contact.jsx
          Destinations.jsx
          Footer.jsx
          Header.jsx
          Hero.jsx
          LambdaFetcher.jsx
          Services.jsx
          Testimonials.jsx
          App.jsx
          Index.jsx
          main.jsx
          vite-env.d.ts
          gllgnore

App.jsx Header.jsx About.jsx Contact.jsx LambdaFetcher.jsx Destinations.jsx Hero.jsx Footer.jsx

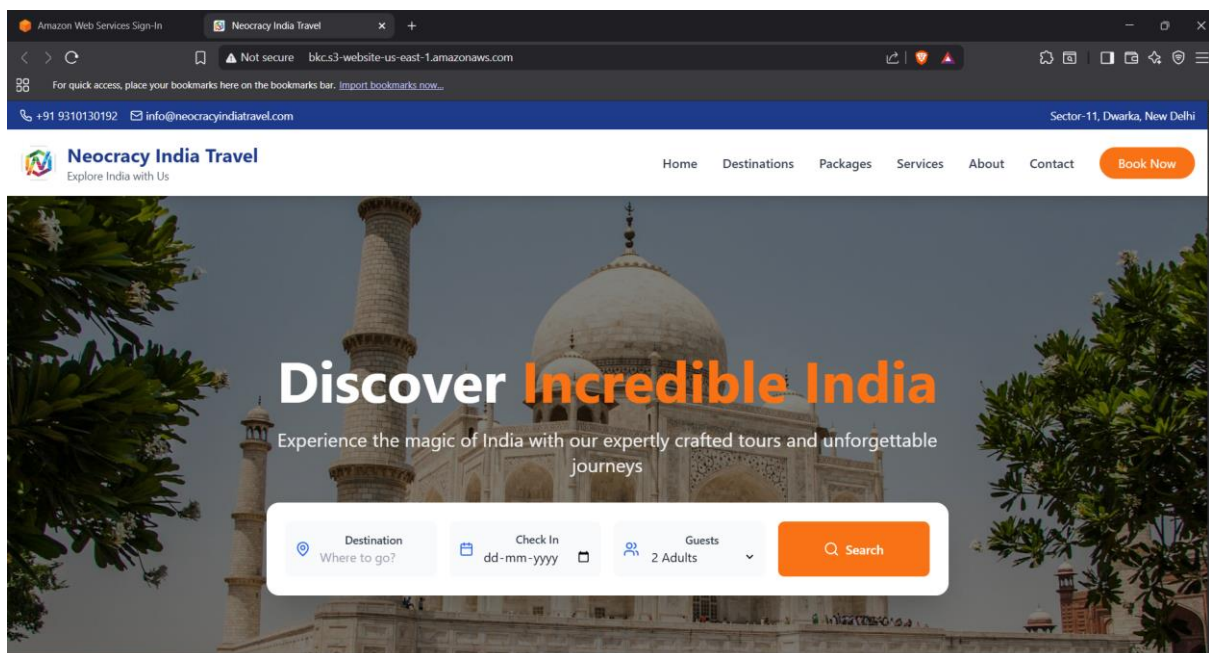
1 import React, { useState } from 'react';
2 import { Phone, Mail, MapPin, Clock, Send } from 'lucide-react';
3 import axios from 'axios';
4
5 const Contact = () => {
6   const [formData, setFormData] = useState({
7     name: '',
8     email: '',
9     phone: '',
10    destination: '',
11    message: ''
12  });
13
14  const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement | HTMLSelectElement>) => {
15    setFormData({
16      ...formData,
17      [e.target.name]: e.target.value
18    });
19  };
20
21  const handleSubmit = async (e: React.FormEvent) => {
22    e.preventDefault();
23    try {
24      const res = await axios.post('https://och25lp0nb.execute-api.us-east-1.amazonaws.com/contact', formData);
25      console.log('Response:', res.data);
26      alert('Submitted successfully!');
27    } catch (error) {
28      console.error('Error submitting form:', error);
29      alert('Something went wrong.');

```

3. Built production files using npm run build and uploaded the /dist folder to an **AWS S3 bucket** with public read permissions



4. Enabled static website hosting on the S3 bucket and tested the live URL



## 5. AWS Lambda Function

- Created a Python 3 Lambda function
- Code parses incoming form data, formats it, and triggers two actions:
  - Publishes an **SNS notification** with form details
  - Stores data in **DynamoDB** with a timestamp and unique ID

aws lambda\_function | Functions | Lambda | Neocracy India Travel

us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1

Search [Alt+S] United States (N. Virginia) jhotta

Lambda > Functions > lambda\_function

### lambda\_function

Throttle Copy ARN Actions

Function overview Info

Export to Infrastructure Composer Download

Diagram Template

lambda\_function

Layers (0)

API Gateway (2)

SNS

+ Add trigger

+ Add destination

Description

Last modified 2 days ago

Function ARN  
arn:aws:lambda:us-east-1:929075265182:function:lambda\_function

Function URL Info

aws lambda\_function | Functions | Lambda | Neocracy India Travel

us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1

Search [Alt+S] United States (N. Virginia) jhotta

Lambda > Functions > lambda\_function

### Code source Info

Open in Visual Studio Code Upload from

lambda\_function

EXPLORER

LAMBDA\_FUNCTION

lambda\_function.py

```
1 import json
2 import boto3
3 import uuid
4 from datetime import datetime
5
6 sns = boto3.client('sns')
7 dynamodb = boto3.resource('dynamodb')
8 table = dynamodb.Table('ContactSubmissions')
9 SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:929075265182:mytopic'
10
11 def lambda_handler(event, context):
12     try:
13         body = json.loads(event.get('body', '{}'))
14
15         name = body.get('name', 'N/A')
16         email = body.get('email', 'N/A')
17         phone = body.get('phone', 'N/A')
18         destination = body.get('destination', 'N/A')
19         message = body.get('message', 'N/A')
20
21         # Save to DynamoDB
22         table.put_item(Item={
23             'id': str(uuid.uuid4()),
24             'name': name,
25             'email': email,
26             'phone': phone,
27             'destination': destination,
28             'message': message,
29             'timestamp': datetime.utcnow().isoformat()
30         })
31     except Exception as e:
32         print(f'Error: {e}')
33         return {'statusCode': 500, 'body': 'Internal Server Error'}
```

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+T)

TEST EVENTS (SELECTED: T...)

Create new test...

Private saved ev...

test-event

ENVIRONMENT VARIABLES

Amazon Q

Ln 12, Col 9 Spaces: 4 UTF-8 LF Python Lambda Layout US

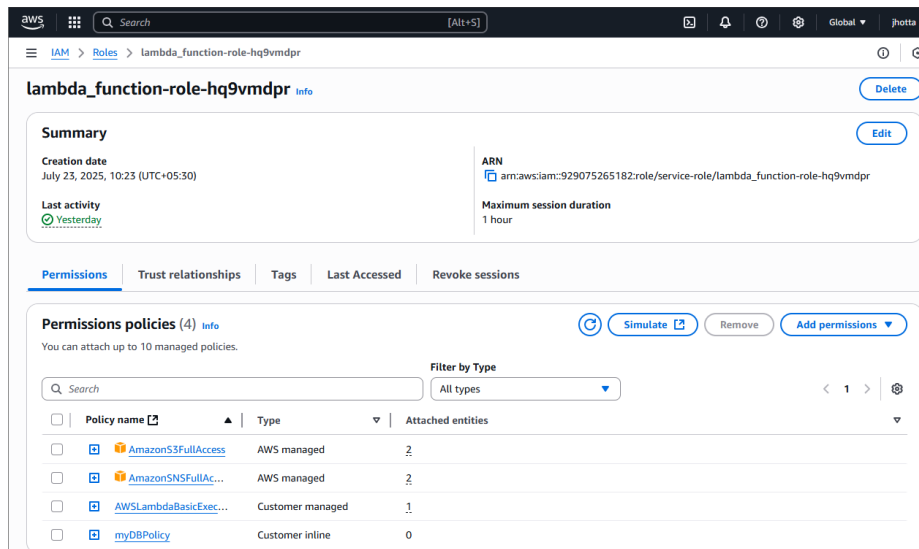
The screenshot shows the AWS Lambda console's 'Code source' tab for a function named 'lambda\_function'. The interface includes a left sidebar with 'EXPLORER' and 'DEPLOY' sections. The main area displays a Python code editor for 'lambda\_function.py'. The code defines a 'lambda\_handler' function that composes an SMS message from event data and publishes it to an SNS topic. The message subject is 'New Travel Query from Neocracy India Travel'. The function returns a 200 status code and a JSON body indicating successful submission. The bottom status bar shows 'Ln 20, Col 1, Spaces: 4, UTF-8, LF, Python, Lambda, Layout: US'.

```
11 def lambda_handler(event, context):
12     try:
13         # Compose message
14         sns_message = f"""
15         New Contact Form Submission:
16
17         Name: {name}
18         Email: {email}
19         Phone: {phone}
20         Destination: {destination}
21         Message: {message}
22         """
23
24         # Publish to SNS
25         response = sns.publish(
26             TopicArn=SNS_TOPIC_ARN,
27             Message=sns_message,
28             Subject="New Travel Query from Neocracy India Travel"
29         )
30
31     except Exception as e:
32         return {"statusCode": 500, "body": str(e)}
```

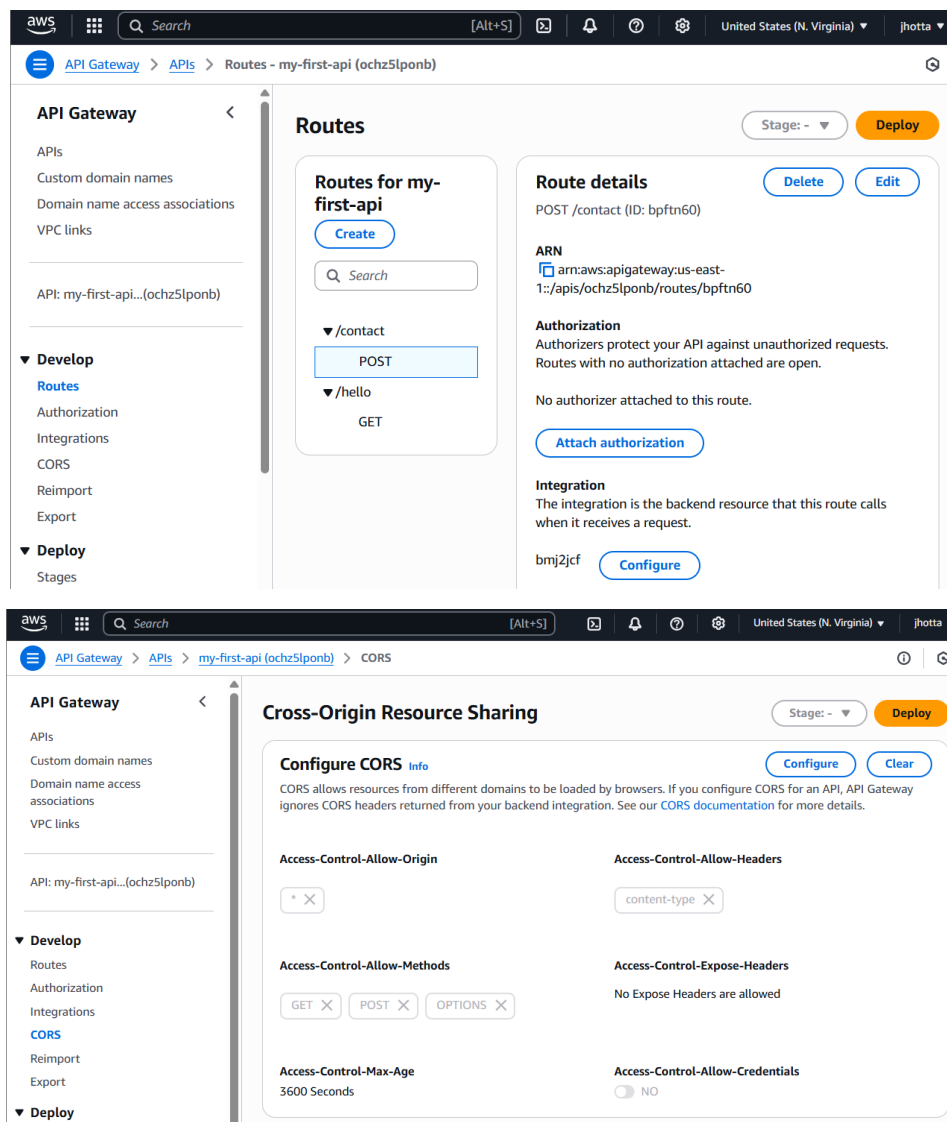
The screenshot shows the 'Configuration' tab of the AWS Lambda console for 'lambda\_function'. The 'Execution role' section is active, displaying the role name 'lambda\_function-role-hq9vmdpr'. Under the 'Resource summary' section, 'AWS End User Messaging SMS and Voice V2' is selected, showing 12 actions and 1 resource. The 'By resource' tab is selected, displaying a list of permissions:

Resource	Actions
	Allow: sms-voice:DescribeVerifiedDestinationNumbers
	Allow: sms-voice:SendDestinationNumberVerificationCode
	Allow: sms-voice>DeleteVerifiedDestinationNumber
	Allow: sms-voice:SetTextMessageSpendLimitOverride
	Allow: sms-voice>DeleteOptedOutNumber

## 6. IAM Role Permissions



## 7. API Gateway Configuration



## 8. SNS Setup



aws

Search

[Alt+S]

United States (N. Virginia)

jhotta

Amazon SNS

Topics

mytopic

mytopic

Edit

Delete

Publish message

Details

Name

mytopic

Display name

-

ARN

arn:aws:sns:us-east-1:929075265182:mytopic

Topic owner

929075265182

Type

Standard

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Encryption

Subscriptions (2)

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

Search

< 1 >

ID

Endpoint

Status

Protocol

a2cd85d4-1953-43...

arn:aws:lambda:us-...

Confirmed

LAMBDA

a4f2b28c-907b-4b...

official.igharjot@g...

Confirmed

EMAIL

## 9. DynamoDB Setup

aws

Search

[Alt+S]

United States (N. Virginia)

jhotta

DynamoDB

Tables

ContactSubmissions

Tables (1)

Any tag key

Any tag value

Find tables

< 1 >

ContactSubmissions

ContactSubmissions

Actions

Explore table items

Settings

Indexes

Monitor

Global tables

Backups

Exports and stre

Protect your DynamoDB table from accidental writes and deletes

Edit PITR

When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 1 to 35 days. Additional charges apply. [Learn more](#)

General information

Info

Get live item count

Partition key

id (String)

Sort key

-

Capacity mode

On-demand

Table status

Active

Alarms

No active alarms

Point-in-time recovery (PITR)

Off

Item count

4

Table size

706 bytes

Average item size

176.5 bytes

Resource-based policy

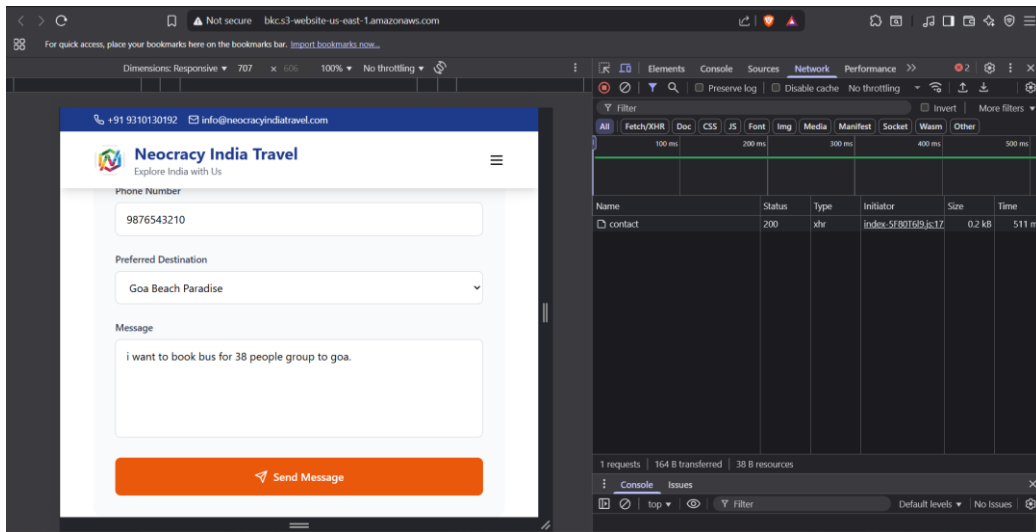
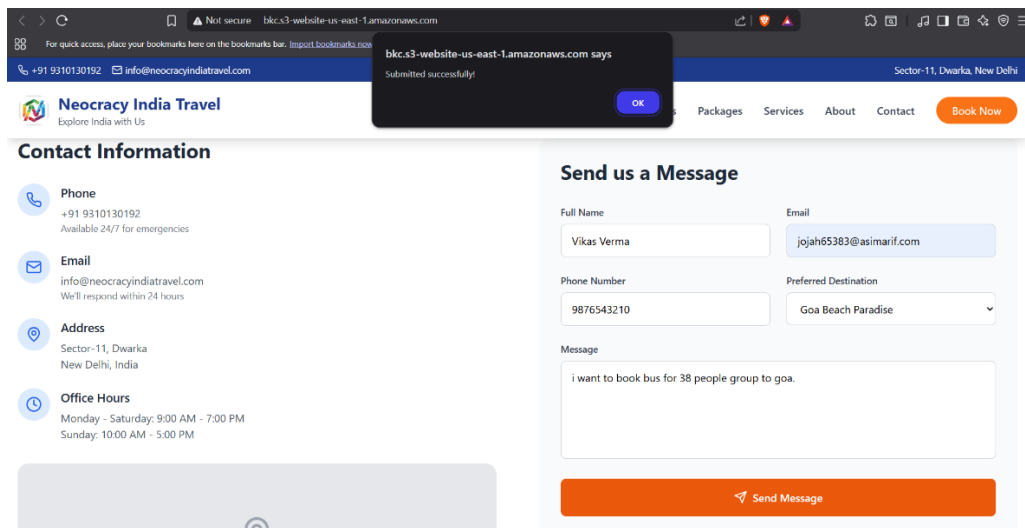
Not active

Amazon Resource Name (ARN)

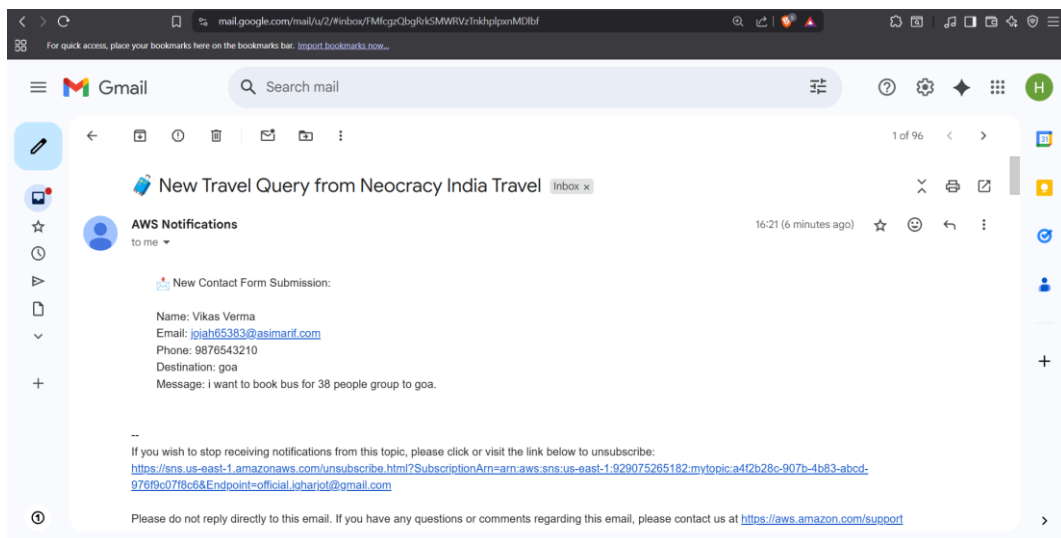
arn:aws:dynamodb:us-east-1:929075265182:table/ContactSubmissions

## 10. Testing & Debugging

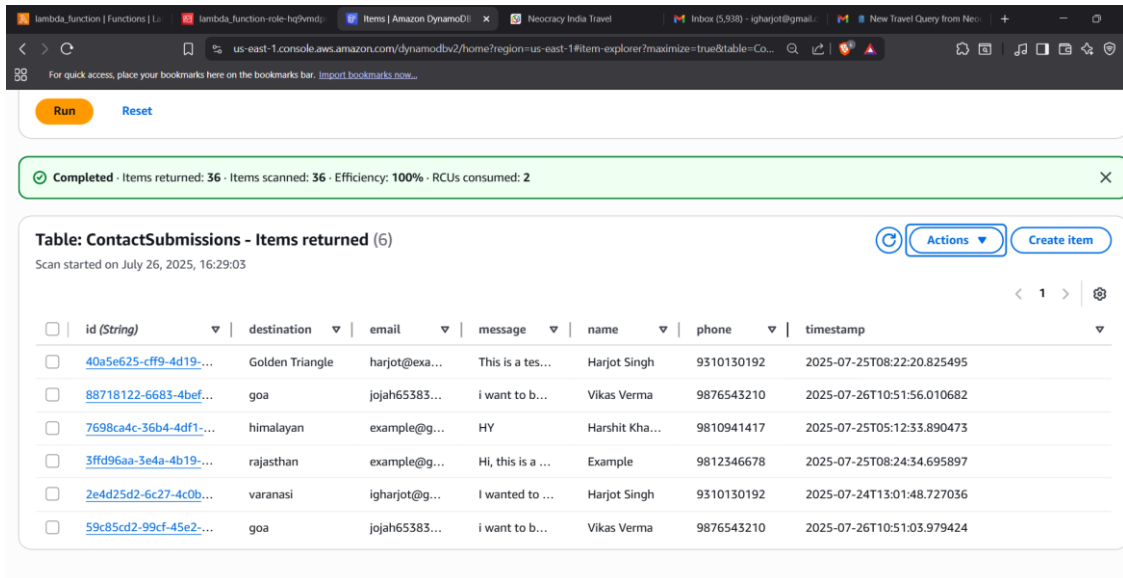
- Used browser DevTools to monitor API requests



- Checked SNS delivery via email inbox



- Confirmed DynamoDB records through AWS Console



Completed · Items returned: 36 · Items scanned: 36 · Efficiency: 100% · RCUs consumed: 2

Table: ContactSubmissions - Items returned (6)  
Scan started on July 26, 2025, 16:29:03

<input type="checkbox"/>	id (String)	destination	email	message	name	phone	timestamp
<input type="checkbox"/>	40a5e625-cff9-4d19-...	Golden Triangle	harjot@exa...	This is a tes...	Harjot Singh	9310130192	2025-07-25T08:22:20.825495
<input type="checkbox"/>	88718122-6683-4bef-...	goa	jojah65383...	i want to b...	Vikas Verma	9876543210	2025-07-26T10:51:56.010682
<input type="checkbox"/>	7698ca4c-36b4-4df1-...	himalayan	example@g...	HY	Harshit Kha...	9810941417	2025-07-25T05:12:33.890473
<input type="checkbox"/>	3ffd96aa-3e4a-4b19-...	rajasthan	example@g...	Hi, this is a ...	Example	9812346678	2025-07-25T08:24:34.695897
<input type="checkbox"/>	2e4d25d2-6c27-4c0b-...	varanasi	igharjot@g...	I wanted to ...	Harjot Singh	9310130192	2025-07-24T13:01:48.727036
<input type="checkbox"/>	59c85cd2-99cf-45e2-...	goa	jojah65383...	i want to b...	Vikas Verma	9876543210	2025-07-26T10:51:03.979424

## Conclusion

**Neocracy Travel Connect** demonstrates the power of serverless architecture using AWS. It eliminates the need for traditional servers while still delivering scalable, secure, and fast performance. This project solidified practical knowledge in deploying React apps on AWS, building Lambda functions, and orchestrating AWS services using IAM, API Gateway, SNS, and DynamoDB.

## Future Enhancements

- Integrate with Google Sheets or export data as CSV
- Build an admin dashboard to manage queries

Harjot Singh  
B.Tech CSE, BPIT, GGSIPU  
GitHub: [github.com/igharjot](https://github.com/igharjot)