# Neocracy Travel Connect

**Harjot Singh and Rishi Pawar**
**Class: CSE-C**
**Roll No: 12320802723 & 11420802723**
**Trainer: Saurabh Dwivedi**

| Assignments | AWS Services used |
|---|---|
| Assignment -1 | EC2, EBS |
| Assignment-2 | EC2, S3, SNS |
| Assignment-3 | RDS, DynamoDB, CloudWatch |
| Assignment-4 | EC2, ELB, Auto-Scaling, Cloud Watch |
| Assignment-5 | Lambda, API Gateway, S3, DynamoDB |

# 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**

---

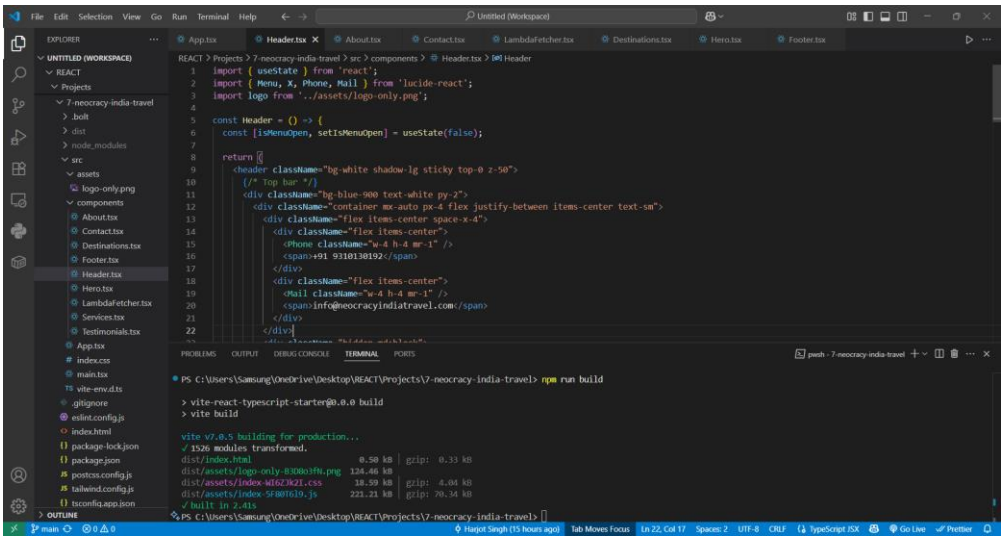
## Technologies Used

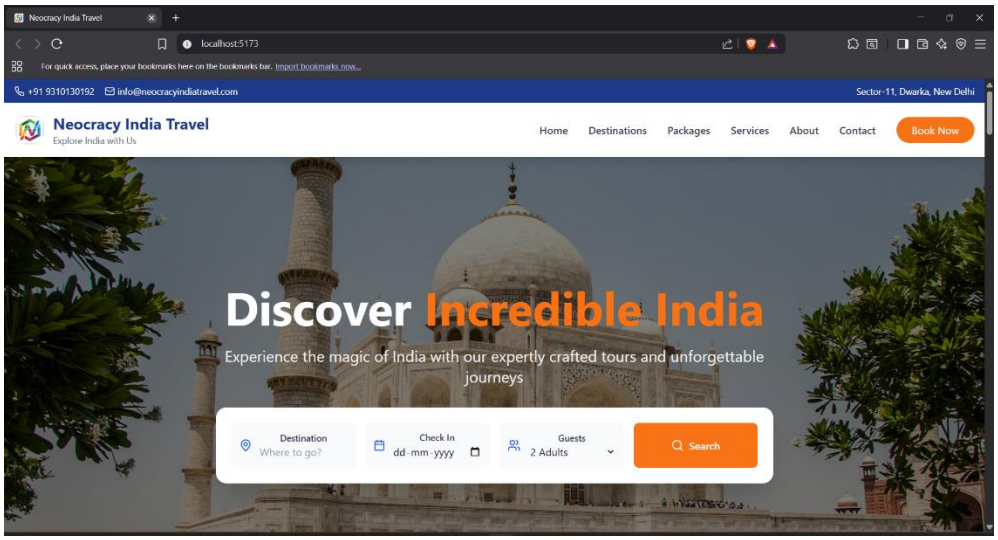
- **Frontend**: React, TypeScript, Tailwind CSS
- **Hosting**: AWS S3 (Static Website Hosting)
- **Backend**: AWS Lambda (Python 3)
- **API Gateway**: For routing frontend requests to Lambda
- **SNS**: For sending real-time email alerts
- **DynamoDB**: For persisting form submissions
- **IAM Roles**: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

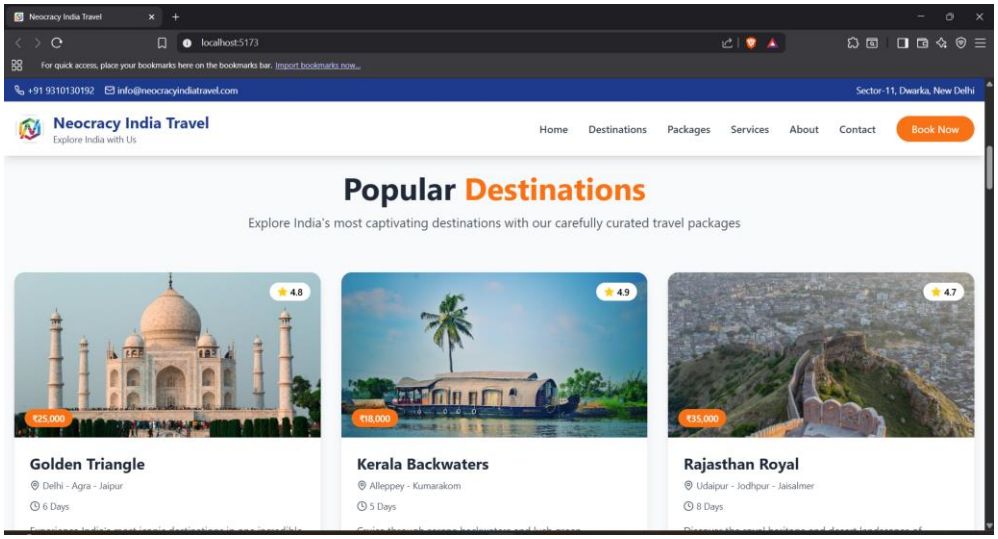
---

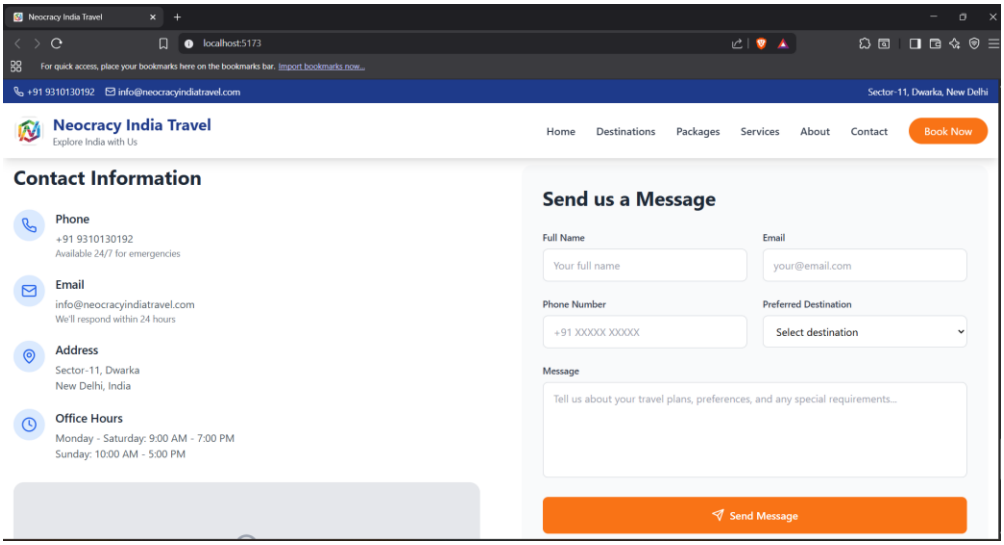
## Implementation Steps:

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

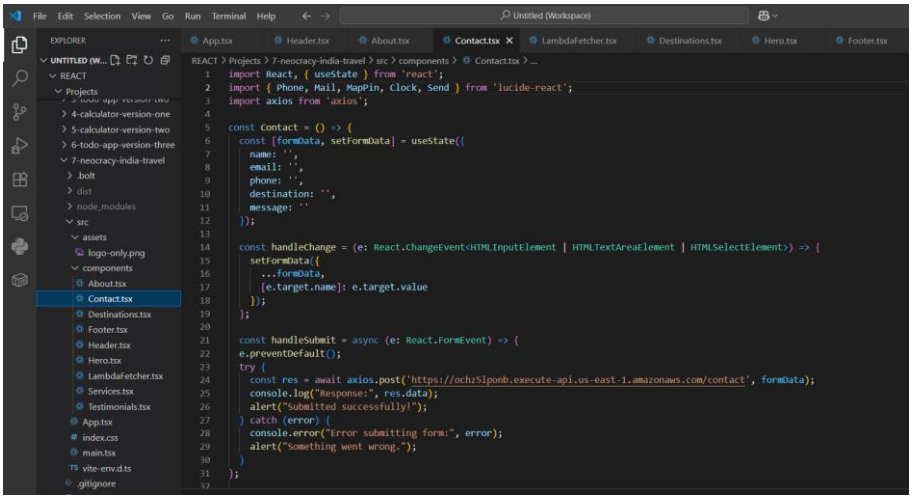


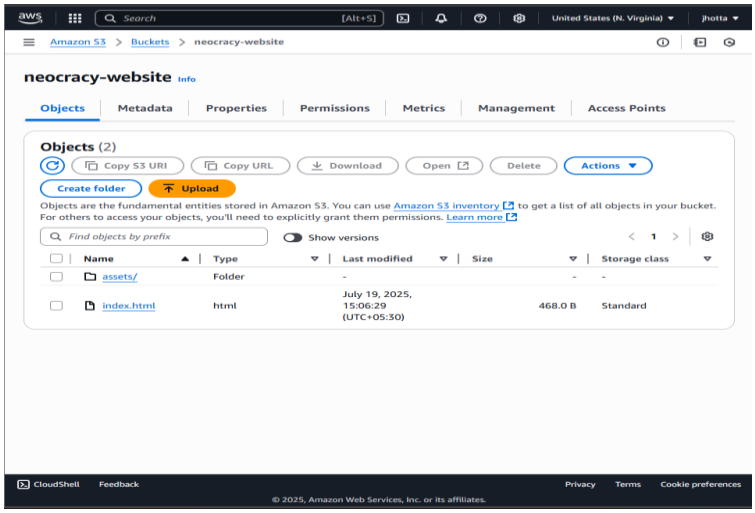





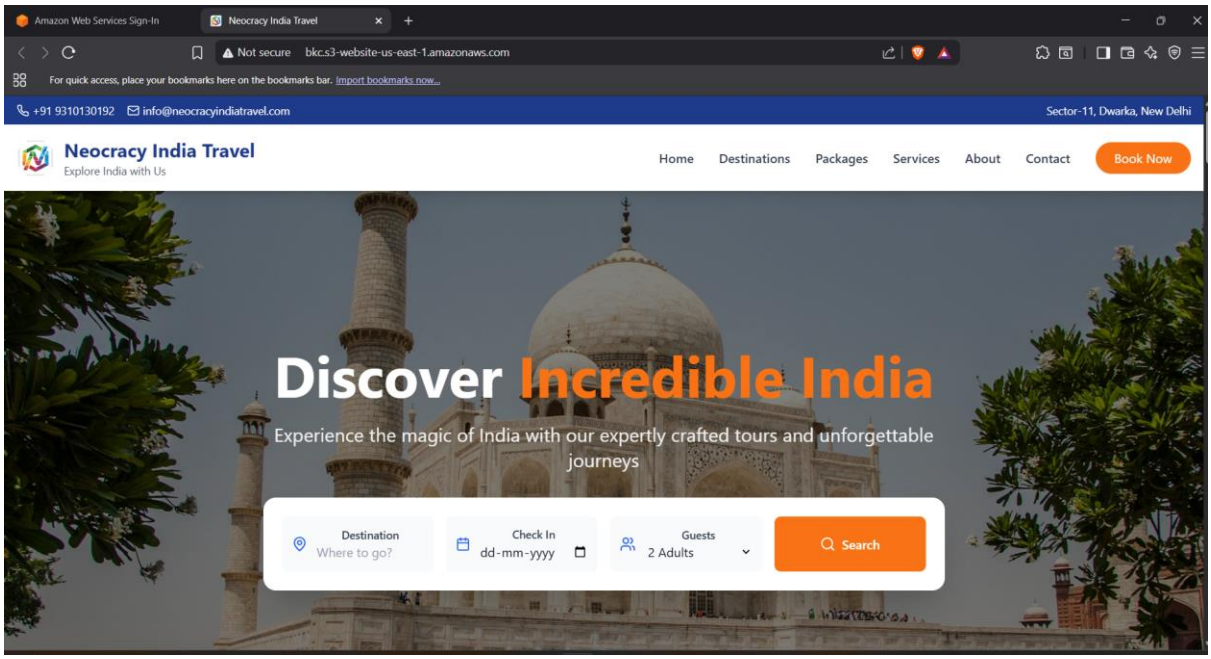

2. Used **Axios** to handle HTTP POST requests from React to API Gateway




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

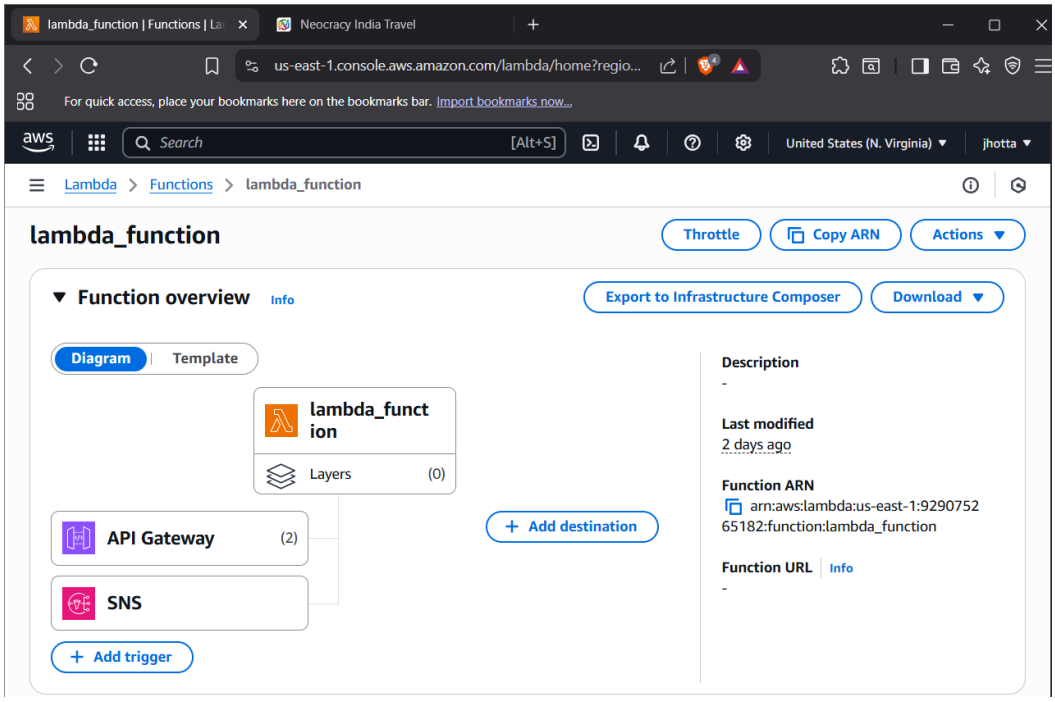


4. **Enabled static website hosting on the S3 bucket and tested the live URL**




## 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



Lambda > Functions > lambda_function

Code source Info

Open in Visual Studio Code | Upload from

lambda_function.py

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

sns = boto3.client('sns')
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('ContactSubmissions')
SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:929075265182:mytopic'

def lambda_handler(event, context):
    try:
        body = json.loads(event.get('body', '{}'))

        name = body.get('name', 'N/A')
        email = body.get('email', 'N/A')
        phone = body.get('phone', 'N/A')
        destination = body.get('destination', 'N/A')
        message = body.get('message', 'N/A')

        # Save to DynamoDB
        table.put_item(Item={
            'id': str(uuid.uuid4()),
            'name': name,
            'email': email,
            'phone': phone,
            'destination': destination,
            'message': message,
            'timestamp': datetime.utcnow().isoformat()
        })
```

Lambda > Functions > lambda_function

Code source Info

Open in Visual Studio Code | Upload from

lambda_function.py

```python
def lambda_handler(event, context):
    try:
        # Compose message
        sns_message = f"""
        📩 New Contact Form Submission:

        Name: {name}
        Email: {email}
        Phone: {phone}
        Destination: {destination}
        Message: {message}
        """

        # Publish to SNS
        response = sns.publish(
            TopicArn=SNS_TOPIC_ARN,
            Message=sns_message,
            Subject='📋 New Travel Query from Neocracy India Travel'
        )

        return {
            "statusCode": 200,
            "headers": {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Headers": "Content-Type",
                "Access-Control-Allow-Methods": "POST, OPTIONS"
            },
            "body": json.dumps({ "message": "Submitted successfully!" })
        }

    except Exception as e:
```

Lambda > Functions > lambda_function

Code | Test | Monitor | Configuration | Aliases | Versions

- General configuration
- Triggers
- Permissions
- Destinations
- Function URL
- Environment variables
- Tags
- VPC
- RDS databases
- Monitoring and operations tools

## Execution role

Edit | View role document

**Role name**

lambda_function-role-hq9vmdpr

### Resource summary

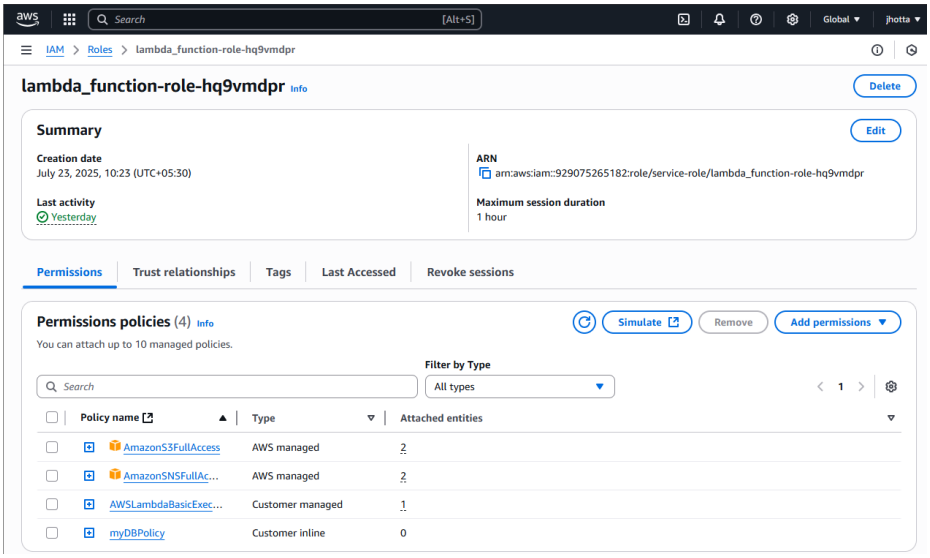
To view the resources and actions that your function has permission to access, choose a service.

AWS End User Messaging SMS and Voice V2
12 actions, 1 resource

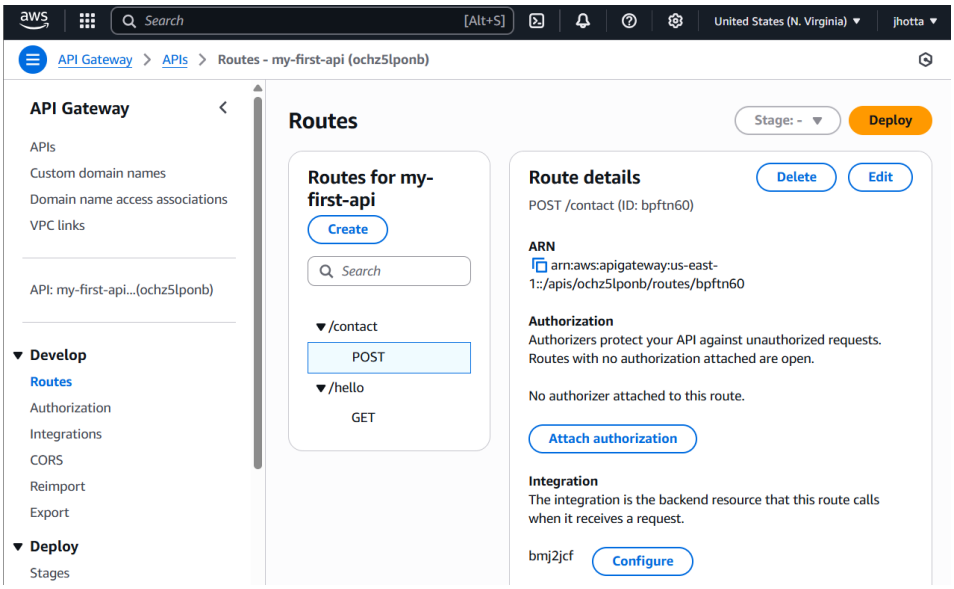
By action | By resource

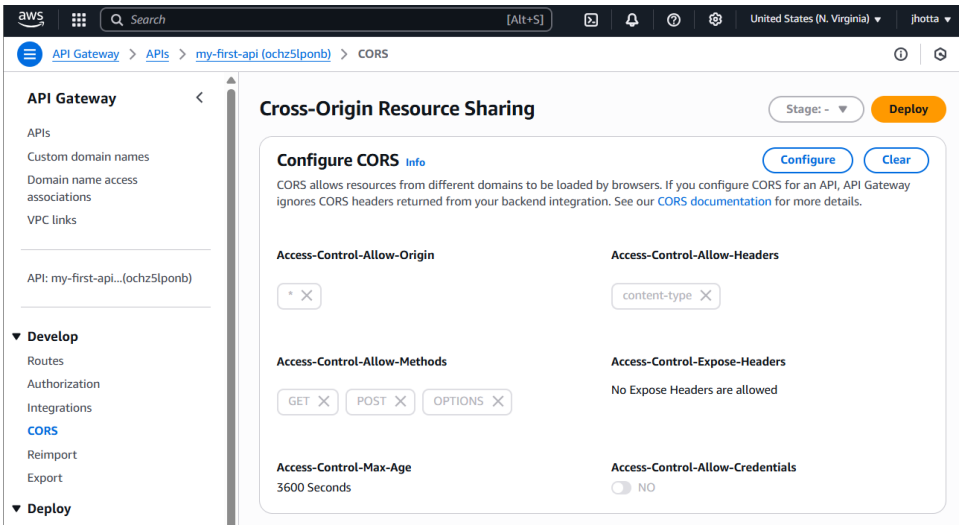
| Resource | Actions |
|---|---|
| | Allow: sms-voice:DescribeVerifiedDestinationNumbers<br>Allow: sms-voice:SendDestinationNumberVerificationCode<br>Allow: sms-voice:DeleteVerifiedDestinationNumber<br>Allow: sms-voice:SetTextMessageSpendLimitOverride<br>Allow: sms-voice:DeleteOptedOutNumber |

## 6. IAM Role Permissions

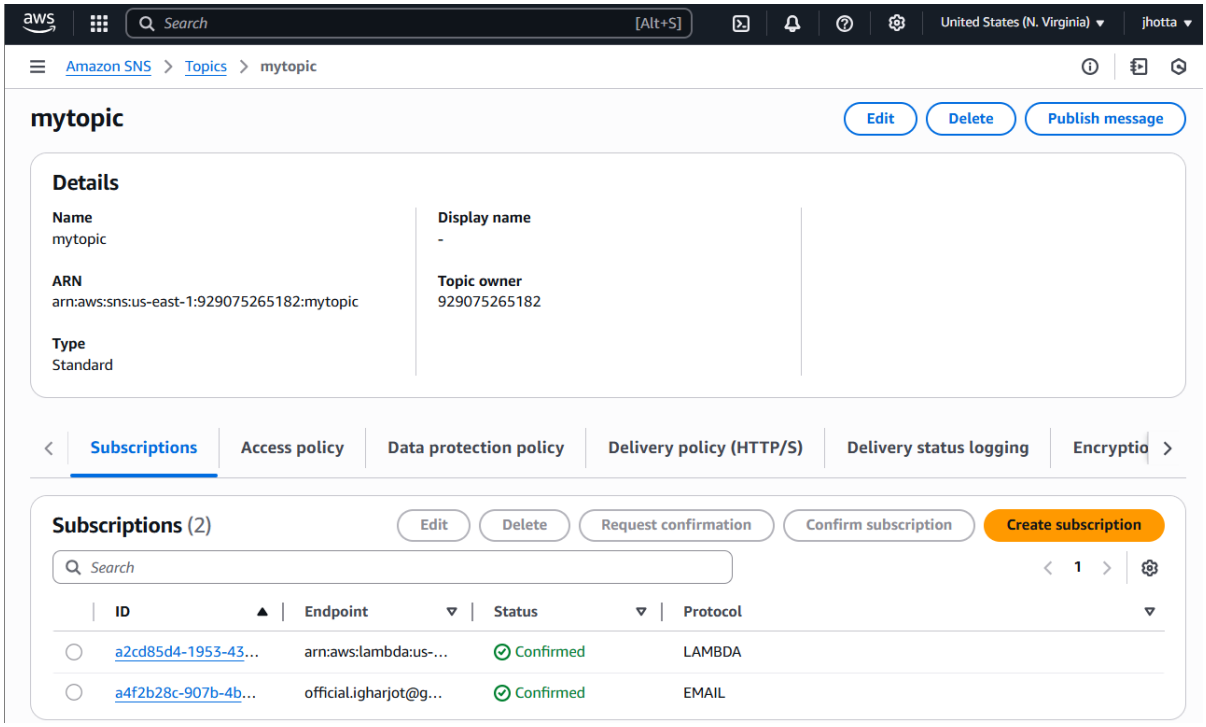



## 7. API Gateway Configuration

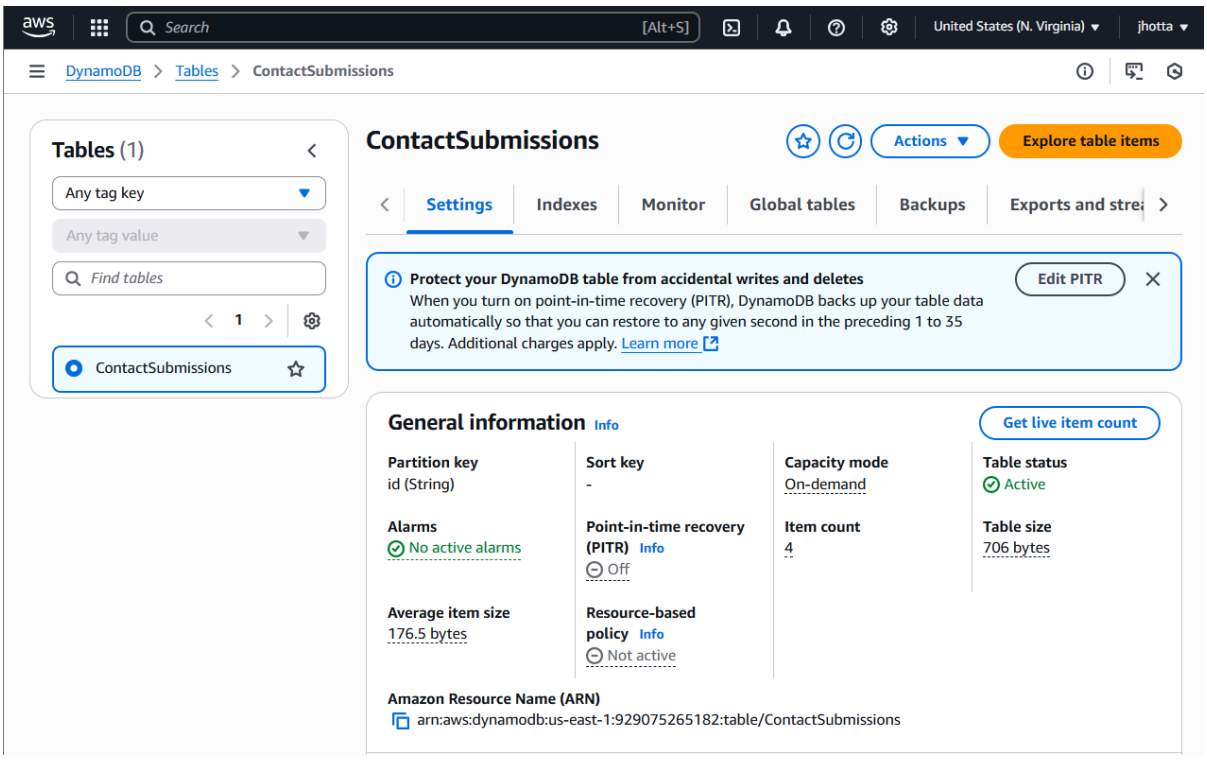


## 8. SNS Setup

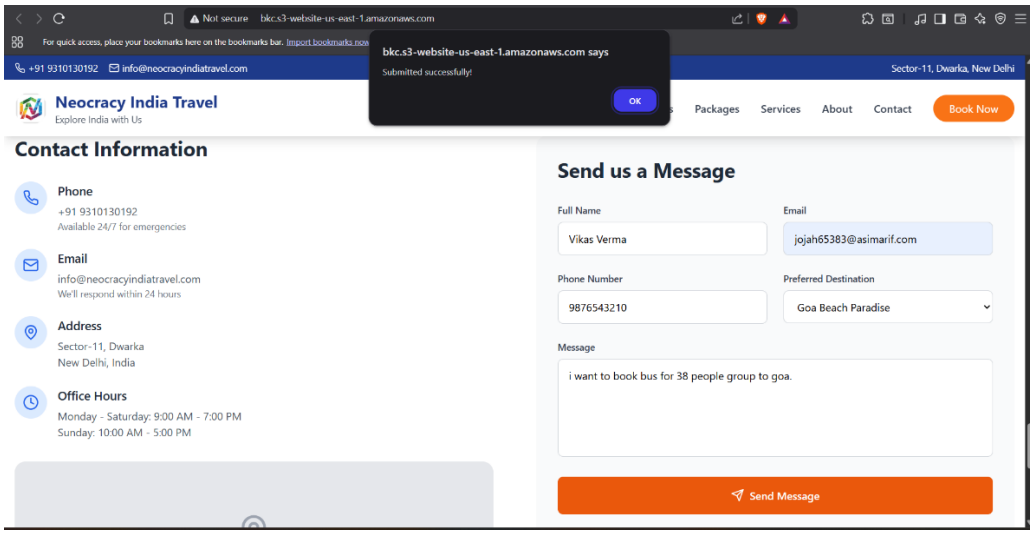


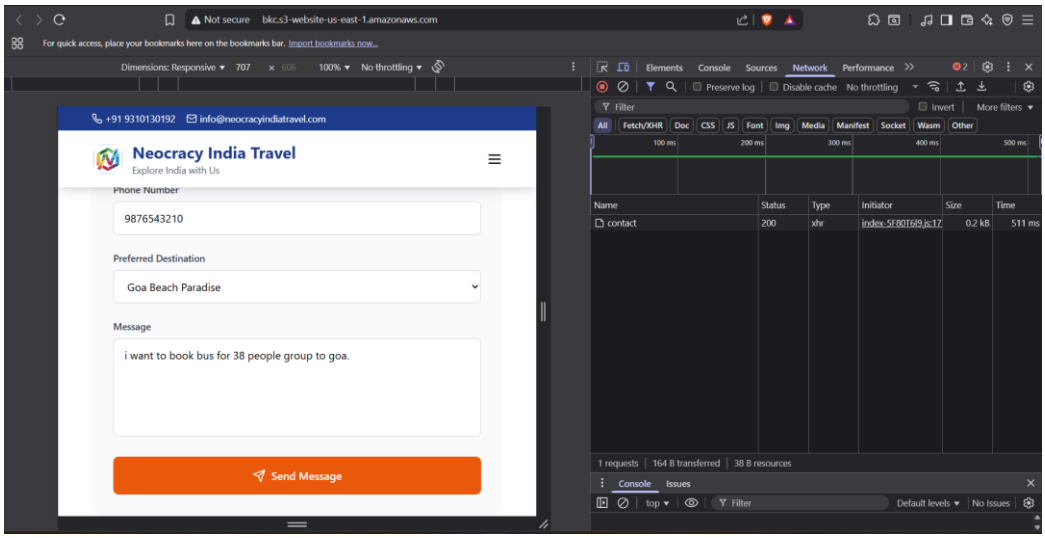

## 9. DynamoDB Setup



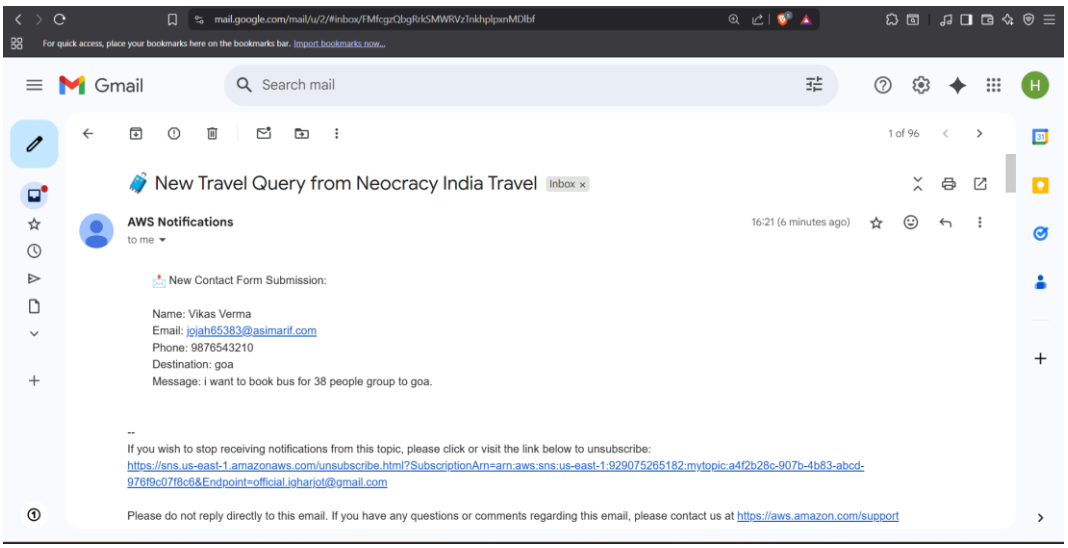
## 10. Testing & Debugging

- **Used browser DevTools to monitor API requests**

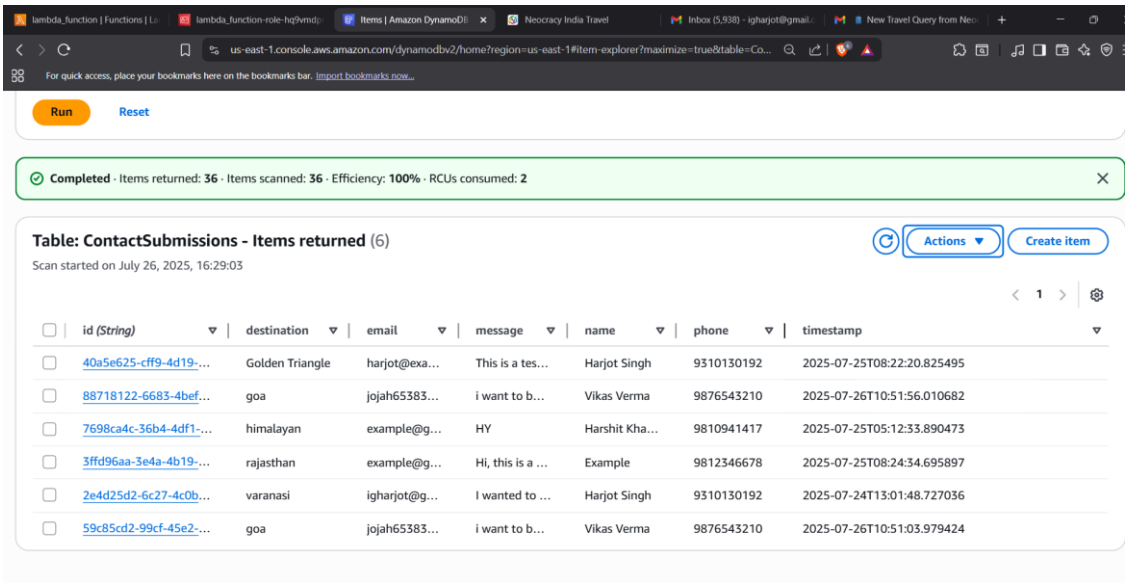



- **Checked SNS delivery via email inbox**



- Confirmed DynamoDB records through AWS Console




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

---

**Project Submitted by:** Harjot Singh and Rishi Pawar
B.Tech CSE, BPIT, GGSIPU
**GitHub:** [github.com/igharjot](https://github.com/igharjot)

**Project Submitted to:** Saurabh Dwivedi