Project-ID-102-FeedbackHub(Web App)

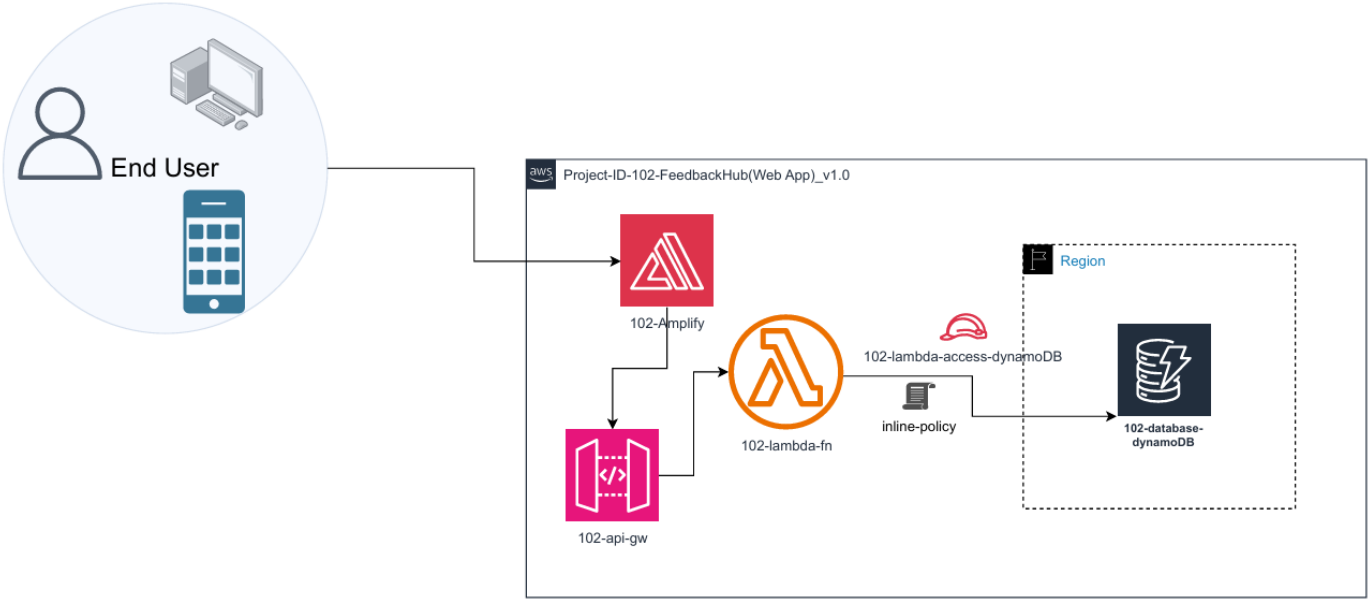
Objective:

This is a simple feedbackHub web app that allows customers or event participants to submit their feedback. The collected feedback will be stored in **DynamoDB** and can be viewed in an **Excel file**.

Architecture flow:
Amplify → API Gateway → Lambda → DynamoDB

AWS Services required:
1. AWS Amplify(Front end)
2. API(connection)
3. Lambda(Back end)
4. DynamoDB(Storage-DB)

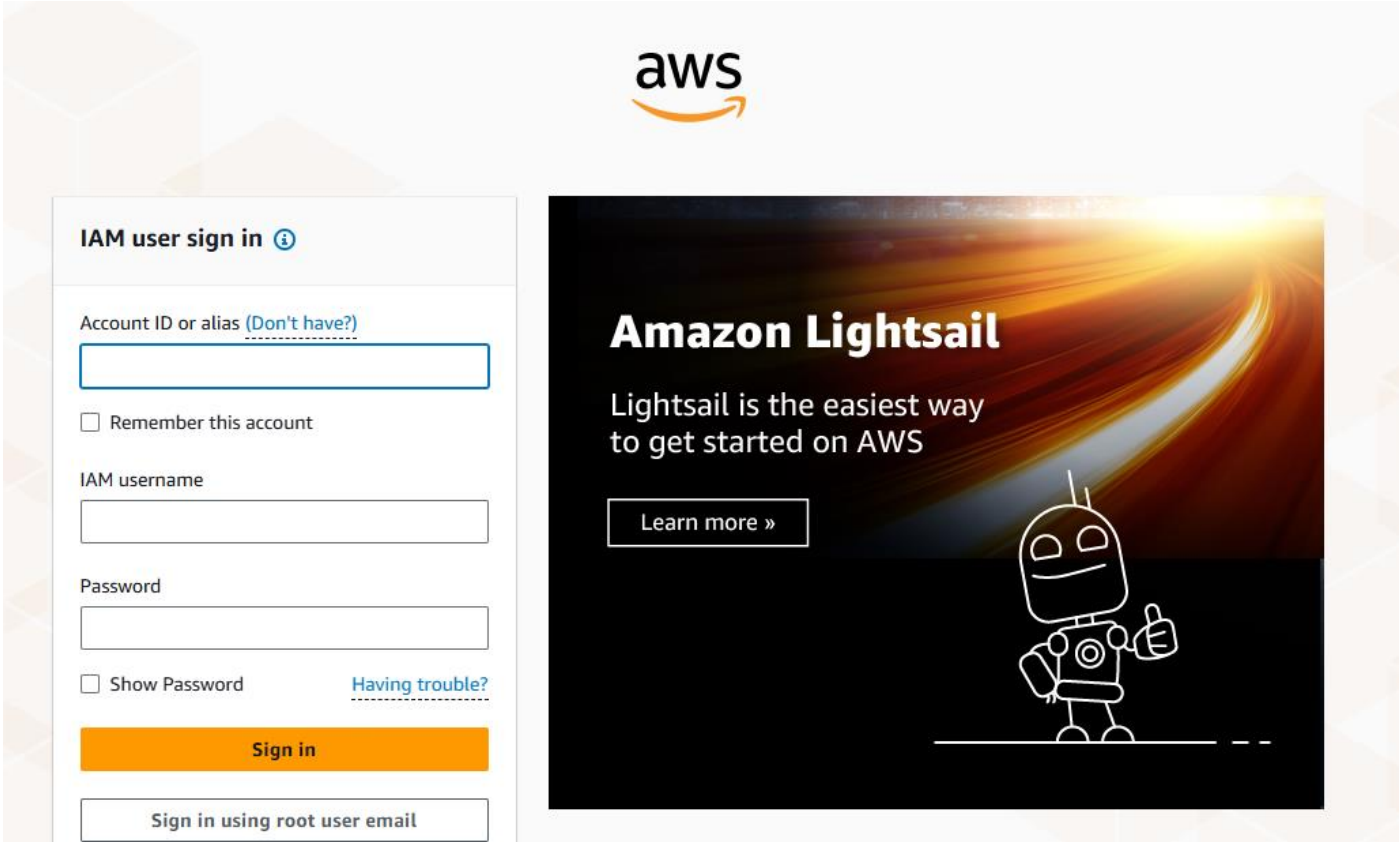
Architecture of FeedbackHub - web App_v1.0



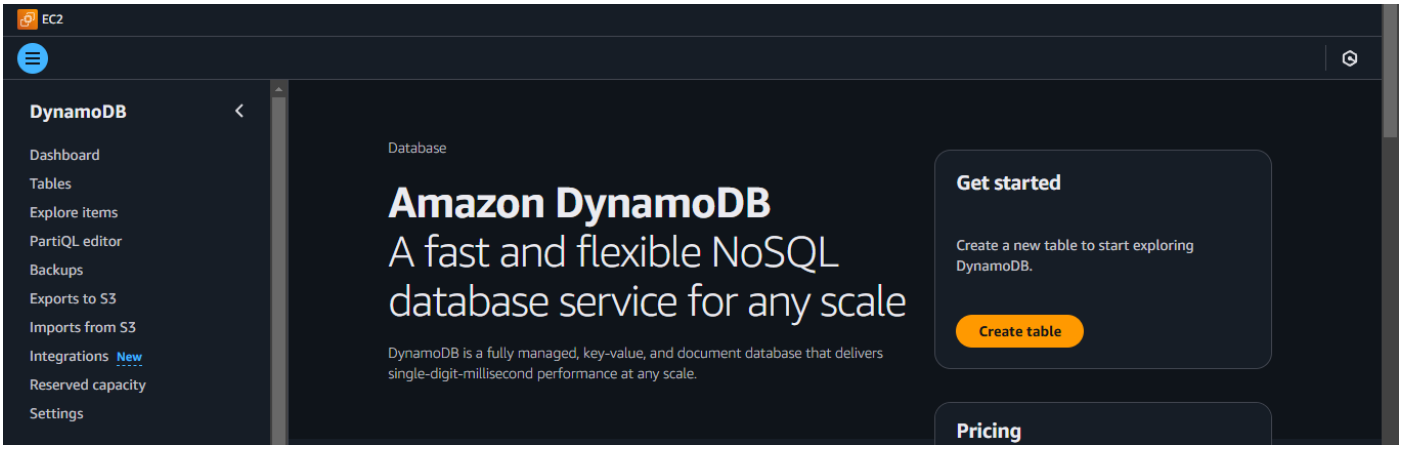
1. Sign In:

1.1 Search https://aws.amazon.com/console/ in browser.

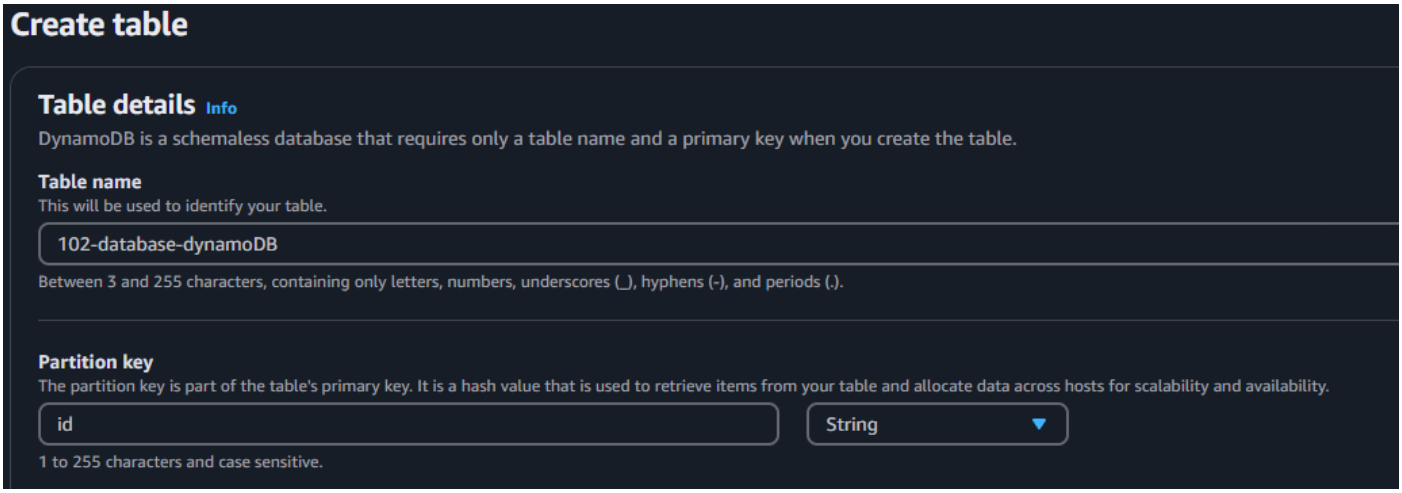
1.2 Sign in if you already have an account or create new and sign in.
(Sign in via IAM user as this a standard and secure procedure)


aws
IAM user sign in
Account ID or alias (Don't have?)
Remember this account
IAM username
Password
Show Password
Having trouble?
Sign in
Sign in using root user email
Amazon Lightsail
Lightsail is the easiest way to get started on AWS
Learn more »

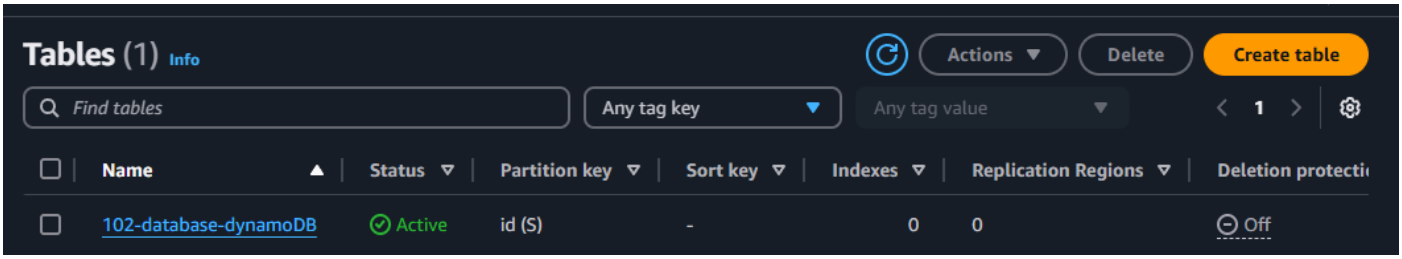
2. Dynamo DB:


EC2
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
Database
Amazon DynamoDB
A fast and flexible NoSQL database service for any scale
DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.
Get started
Create a new table to start exploring DynamoDB.
Create table
Pricing

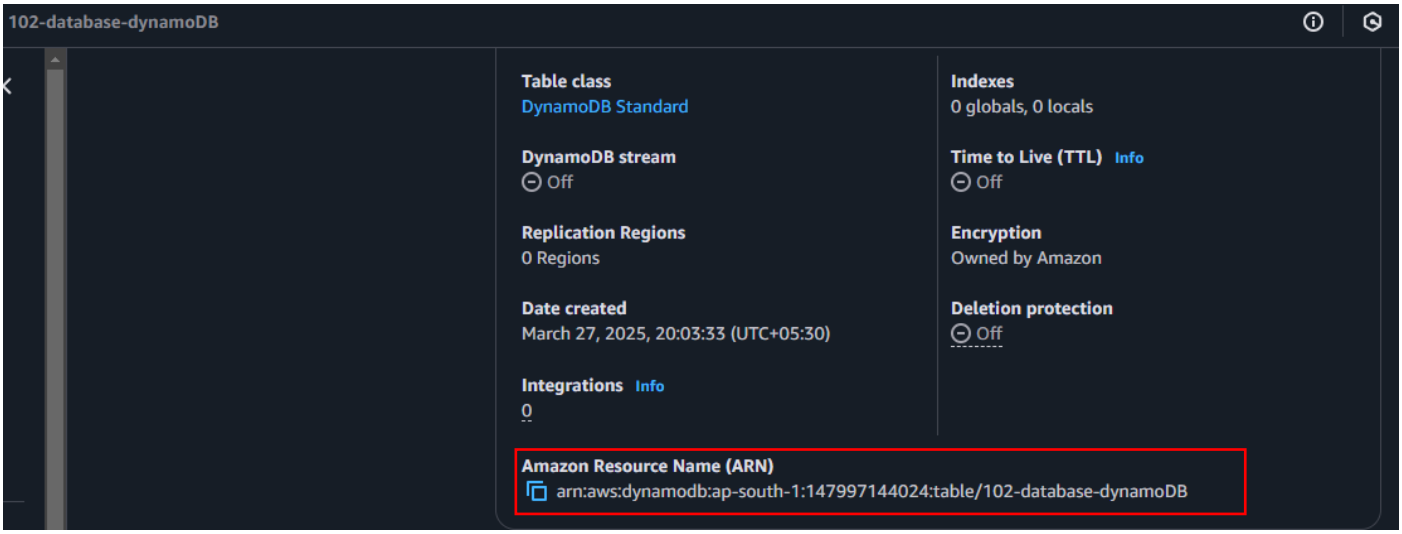
2.1. Create table 102-database-dynamoDB and id as partition key.



2.2 102-database-dynamoDB table has been created.

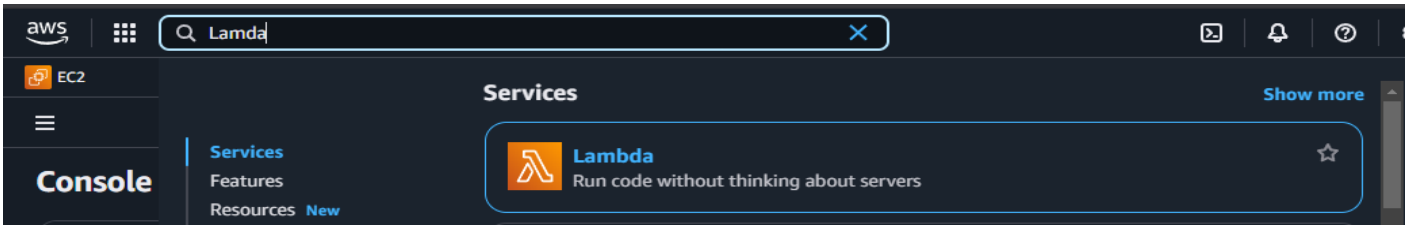


2.3 Note the arn of 102-database-dynamoDB table.

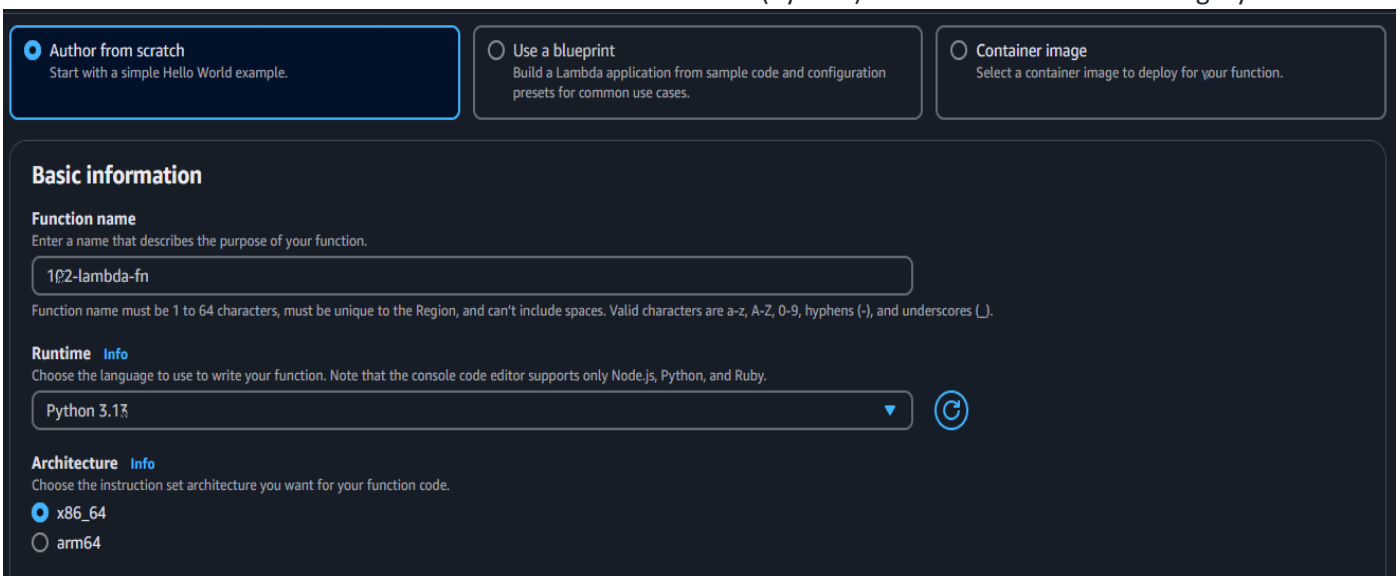


3. AWS Lambda service:

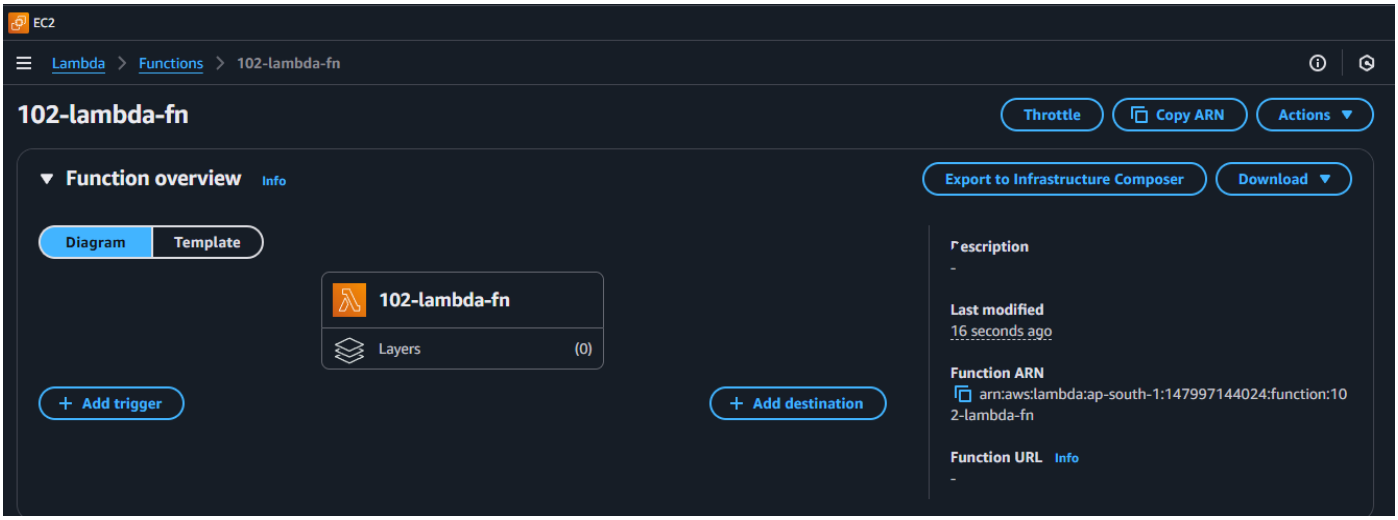
3.1 Go to lambda



3.2 Create a function 102-lambda-fn and choose the runtime (Python) in Author from scratch category.



3.3 102-lambda-fn has been created.

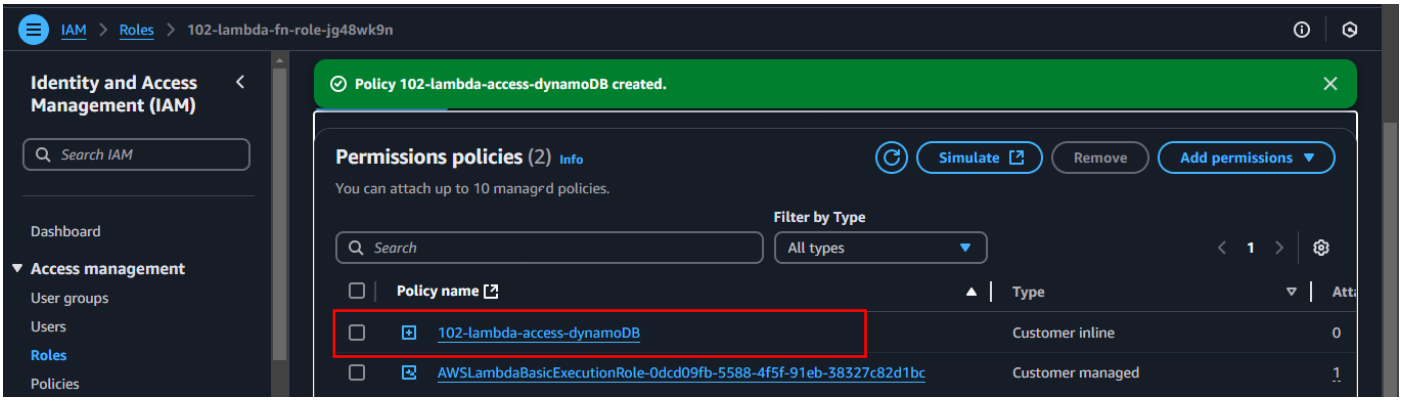


*Add permission for lambda to access dynamoDB

*Go to configuration → Permission → click the role name – (navigate to IAM) → click add permission → create inline permission → JSON → write the below policy & replace the corresponding DynamoDB table arn → give policy name 102-lambda-access-dynamoDB → create policy.

Inline policy

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
          "dynamodb:PutItem",
          "dynamodb:DeleteItem",
          "dynamodb:GetItem",
          "dynamodb:Scan",
          "dynamodb:Query",
          "dynamodb:UpdateItem"
        ],
        "Resource": "YOUR-TABLE-ARN"
      }
      ]
    }
```



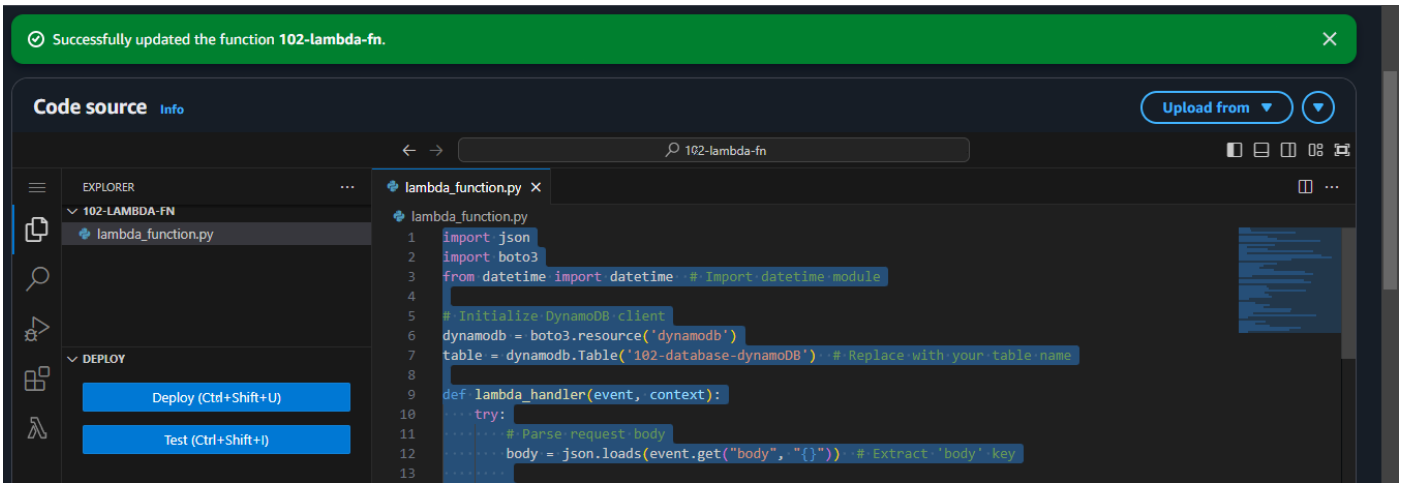
3.4 Write the below code in 102-lamda-fn – (AI generated) – update DynamoDB table name.

```
import json

import boto3

from datetime import datetime  # Import datetime module


# Initialize DynamoDB client

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('102-database-dynamoDB')  # Replace with your table name
```

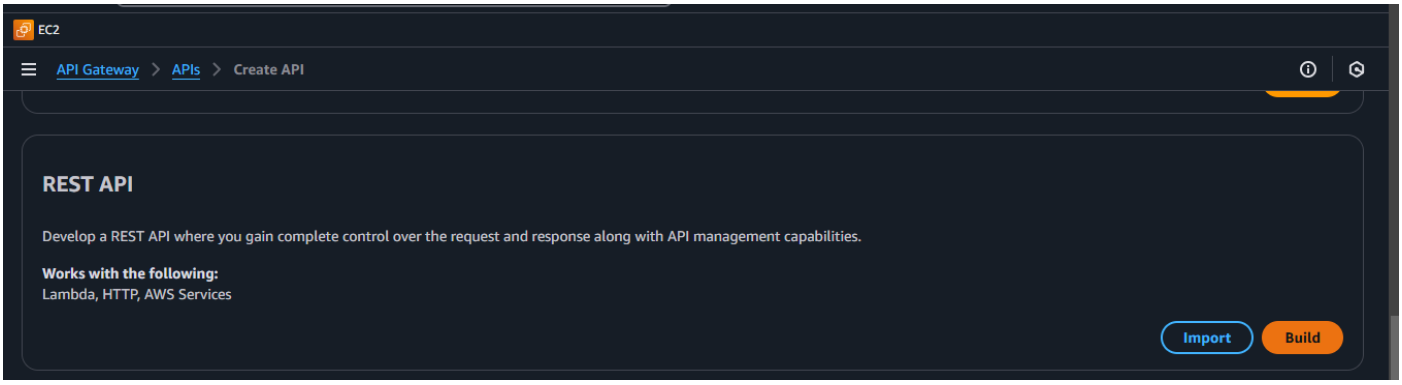
```
def lambda_handler(event, context):
    try:
        # Parse request body
        body = json.loads(event.get("body", "{}"))  # Extract 'body' key

        name = body.get("name", "Anonymous")  # Defaults to "Anonymous" if empty
        mobile = body.get("mobile", "Not Provided")  # Defaults to "Not Provided" if empty
        feedback = body.get("feedback", "").strip()

        # Ensure feedback is provided
        if not feedback:
            return {
                "statusCode": 400,
                "headers": {"Content-Type": "application/json"},
                "body": json.dumps({"message": "Feedback is required!"})
            }

        # Generate formatted timestamp ID (DD/MM/YYYY HH:MM:SS)
        timestamp_id = datetime.utcnow().strftime("%d/%m/%Y %H:%M:%S")

        # Store in DynamoDB
        table.put_item(Item={
            "id": timestamp_id,  # Human-readable timestamp as ID
            "name": name,
            "mobile": mobile,
            "feedback": feedback
        })

        return {
            "statusCode": 200,
```

```
        "headers": {
            "Content-Type": "application/json",
            "Access-Control-Allow-Origin": "*"
        },
        "body": json.dumps({"message": "Feedback successfully saved."})
    }

except Exception as e:
    return {
        "statusCode": 500,
        "headers": {"Content-Type": "application/json"},
        "body": json.dumps({"error": str(e)})
    }
```

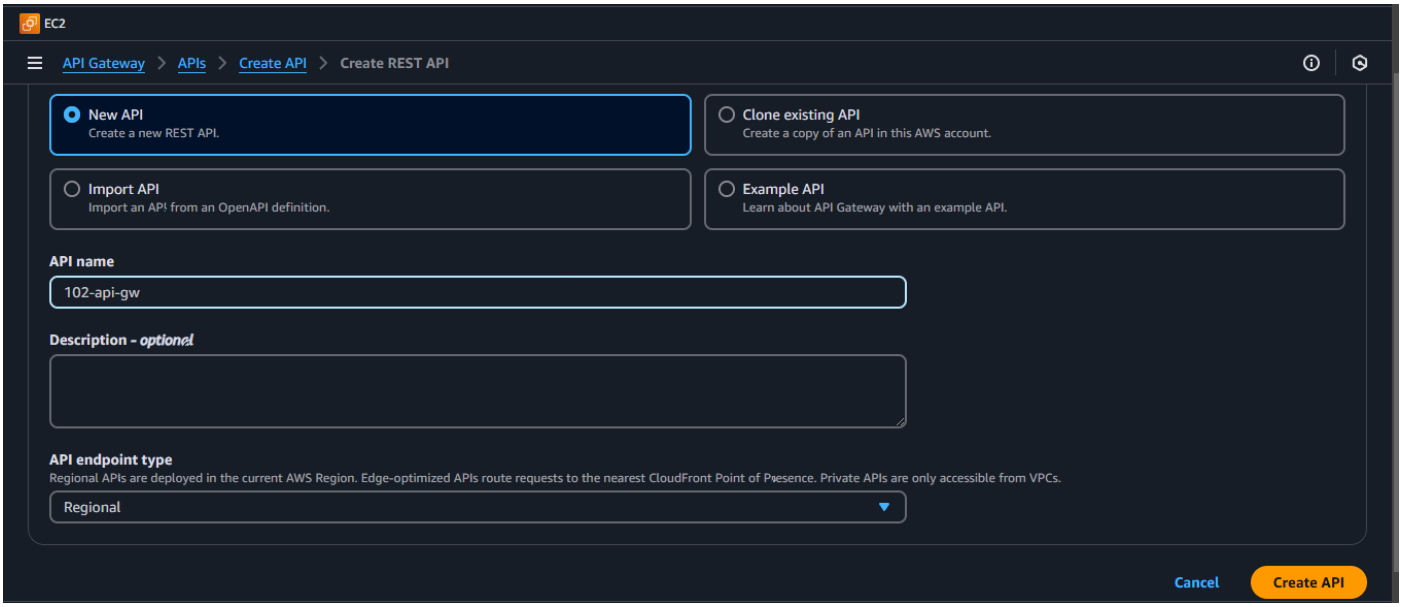
3.5 Deploy the written code in lambda and test.



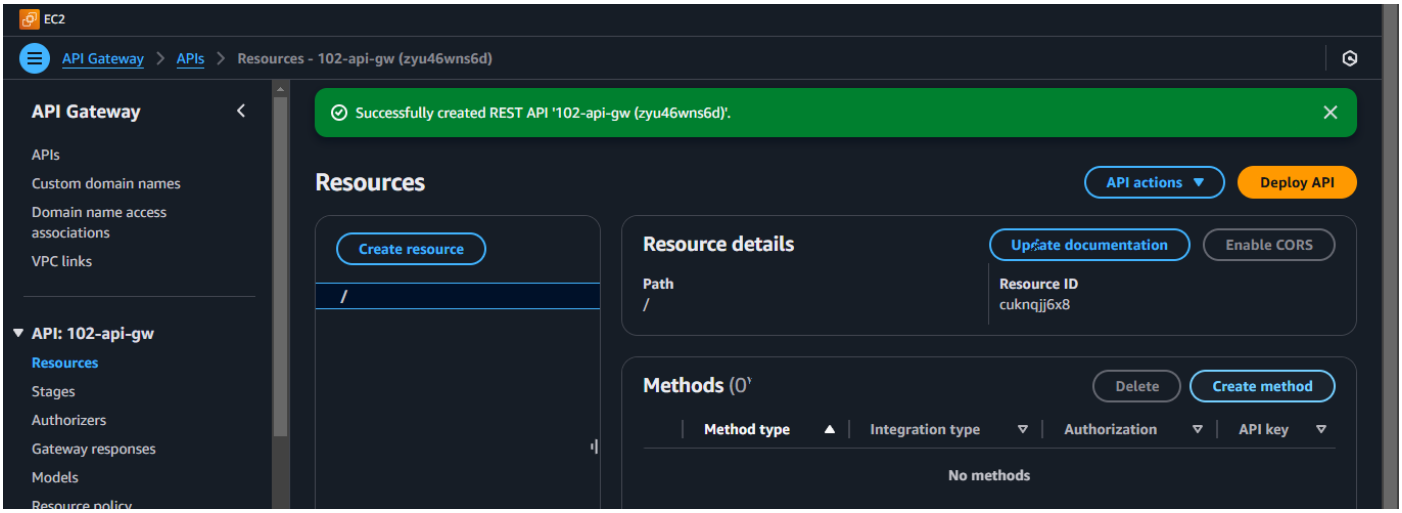
4. Go to API gateway:



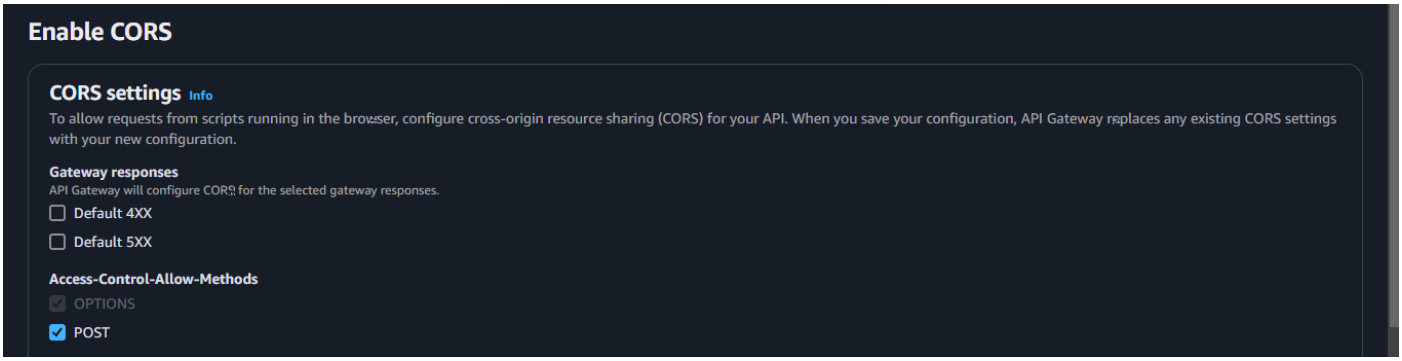
4.1 Choose REST API – 102-api-gw and create it.



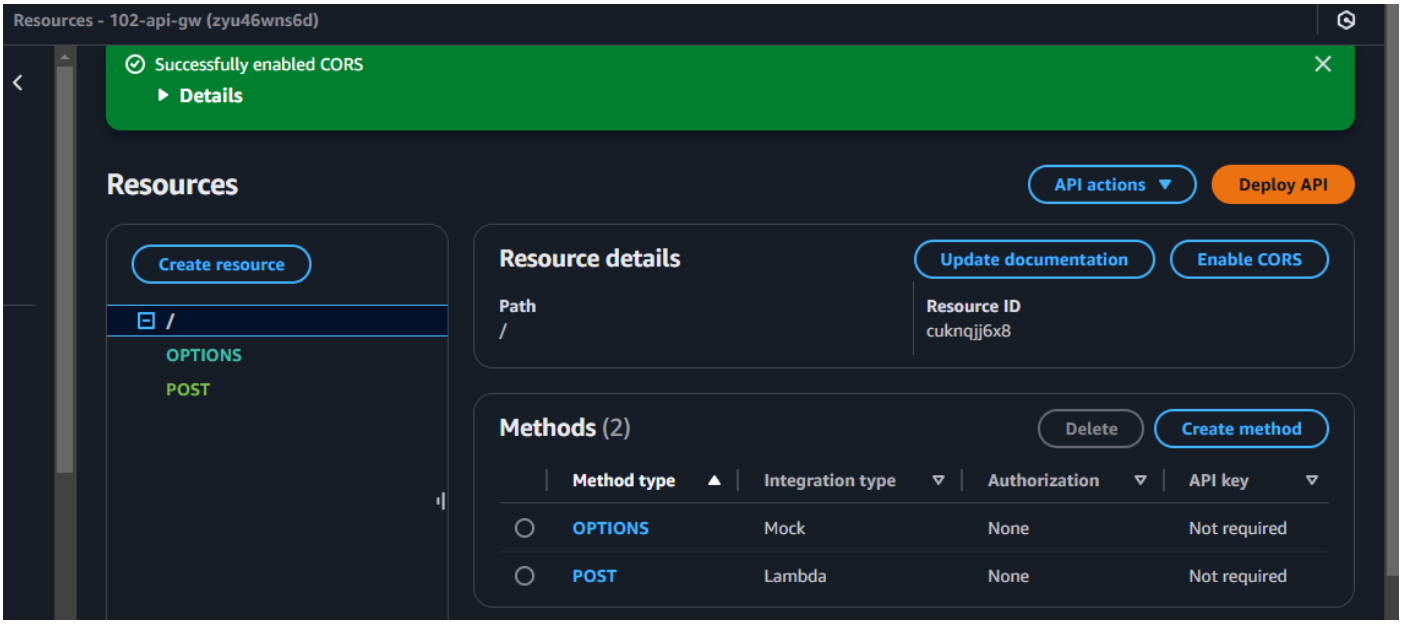
4.2 Create method POST and integration type as lambda and assign arn of 102-lambda-fn.

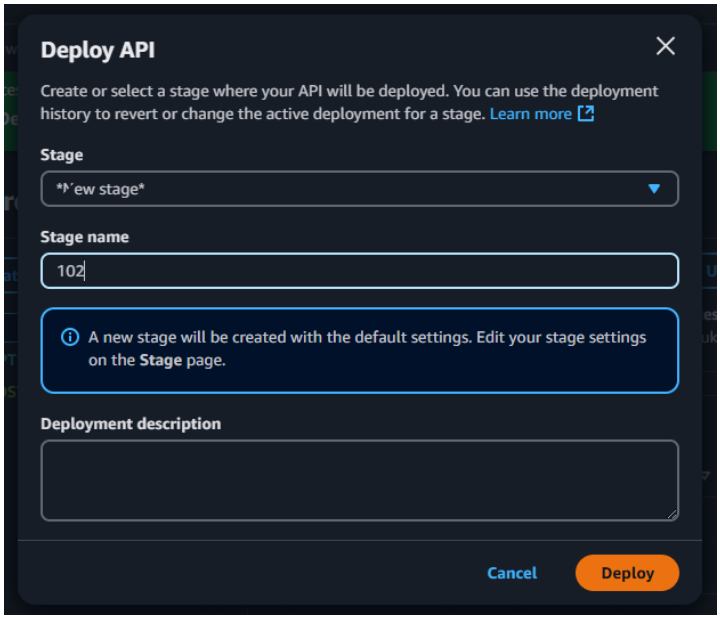


4.3 Refresh and enable the CORS – access control – POST.



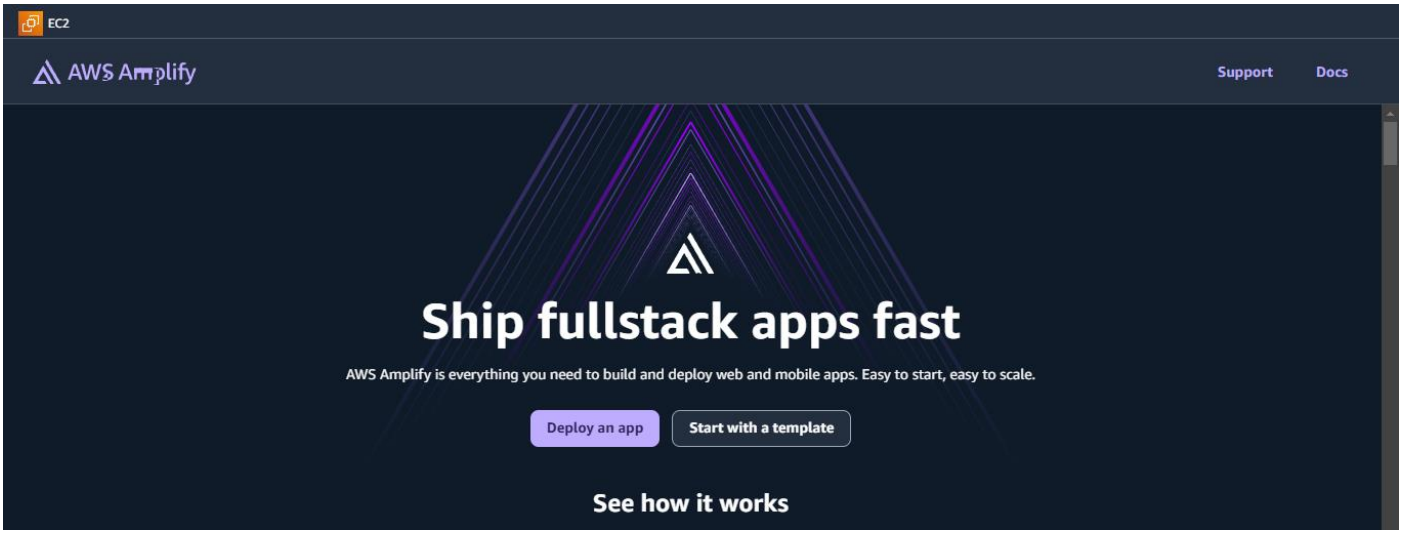
4.3 Deploy the API.





Once deployed - API URL will be generated – update the generated URL in html file.

5. Create Amplify:



*Create the html file – AI generated given below:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Feedback Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #222629;
      color: #d4edda;
      text-align: center;
      margin: 50px;
    }
    .container {
      background: #1e2226;
      padding: 25px;
      border-radius: 10px;
      width: 40%;
```

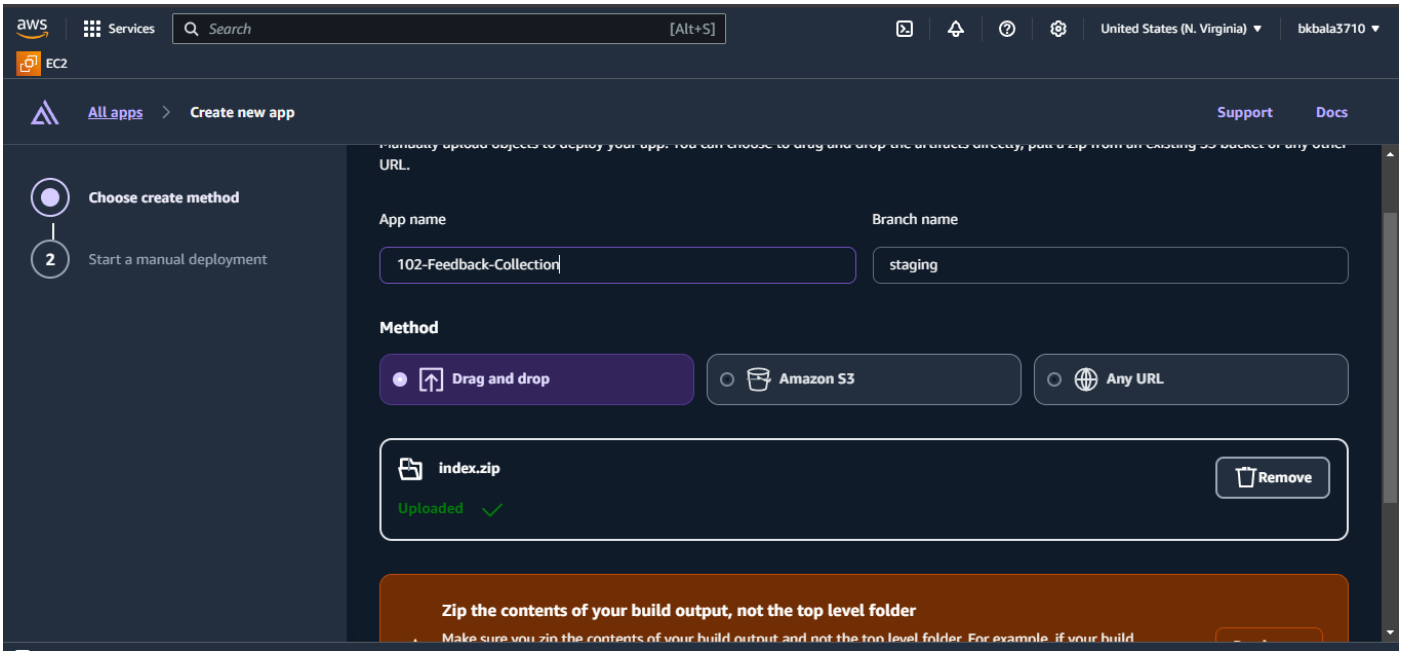
```
    margin: auto;
    border: 2px solid #4caf50; /* Fixed boundary */
}
h2 {
    margin-bottom: 20px;
    font-size: 20px;
    color: #86C232;
}
label {
    font-size: 14px;
    color: #86C232;
    display: block;
    text-align: left;
    margin-left: 5%;
}
input, textarea {
    width: 90%;
    padding: 12px;
    margin: 5px 0 10px;
    border: 1px solid #4caf50;
    border-radius: 5px;
    background-color: #33383d;
    color: #d4edda;
    font-size: 16px;
}
button {
    background-color: #4caf50;
    color: white;
    border: none;
    padding: 12px 18px;
```

```
            font-size: 16px;
            cursor: pointer;
            border-radius: 5px;
            margin-top: 10px;
            width: 95%;
        }
        button:hover {
            background-color: #388e3c;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Dear Participant, thanks for attending - please submit your valuable feedback :) ID 102</h2>
        <form onsubmit="event.preventDefault(); sendFeedback();">
            <label for="name">Name (Optional)</label>
            <input type="text" id="name" placeholder="Your Name">

            <label for="mobile">Mobile Number / Email-ID (Optional)</label>
            <input type="text" id="mobile" placeholder="Your Mobile Number / Email-ID">

            <label for="feedback">Feedback (Required)</label>
            <textarea id="feedback" placeholder="Your Feedback" rows="5" required></textarea><br>

            <button type="submit">Submit Feedback</button>
        </form>
    </div>

    <script>
        function sendFeedback() {
```

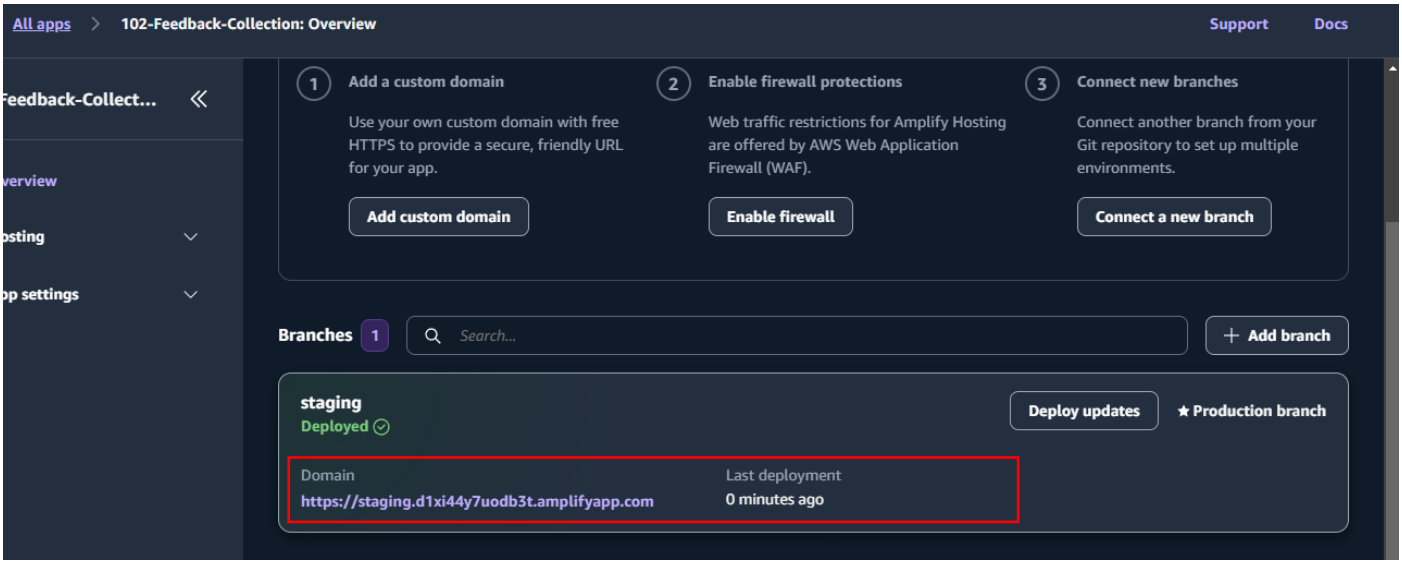
```
var name = document.getElementById('name').value.trim();
var mobile = document.getElementById('mobile').value.trim();
var feedback = document.getElementById('feedback').value.trim();

if (!feedback) {
    alert("Please enter your feedback.");
    return;
}

var apiUrl = "<<API GATEWAY URL>>"; // Replace with your API Gateway URL

var requestData = {
    body: JSON.stringify({
        name: name || "Anonymous",  // Defaults to 'Anonymous' if empty
        mobile: mobile || "Not Provided", // Defaults to 'Not Provided' if empty
        feedback: feedback
    })
};

fetch(apiUrl, {
    method: 'POST',
    headers: {
        "Content-Type": "application/json"
    },
    body: JSON.stringify(requestData)
})
.then(response => response.json())
.then(data => {
    alert("Feedback successfully saved! Thank you.");
    document.getElementById('name').value = "";
```

```
            document.getElementById('mobile').value = "";
            document.getElementById('feedback').value = "";
        })
        .catch(error => console.error('Error:', error));
    }
  </script>
</body>
</html>
```

5.1 Save as html file and convert to zip and upload in Amplify.

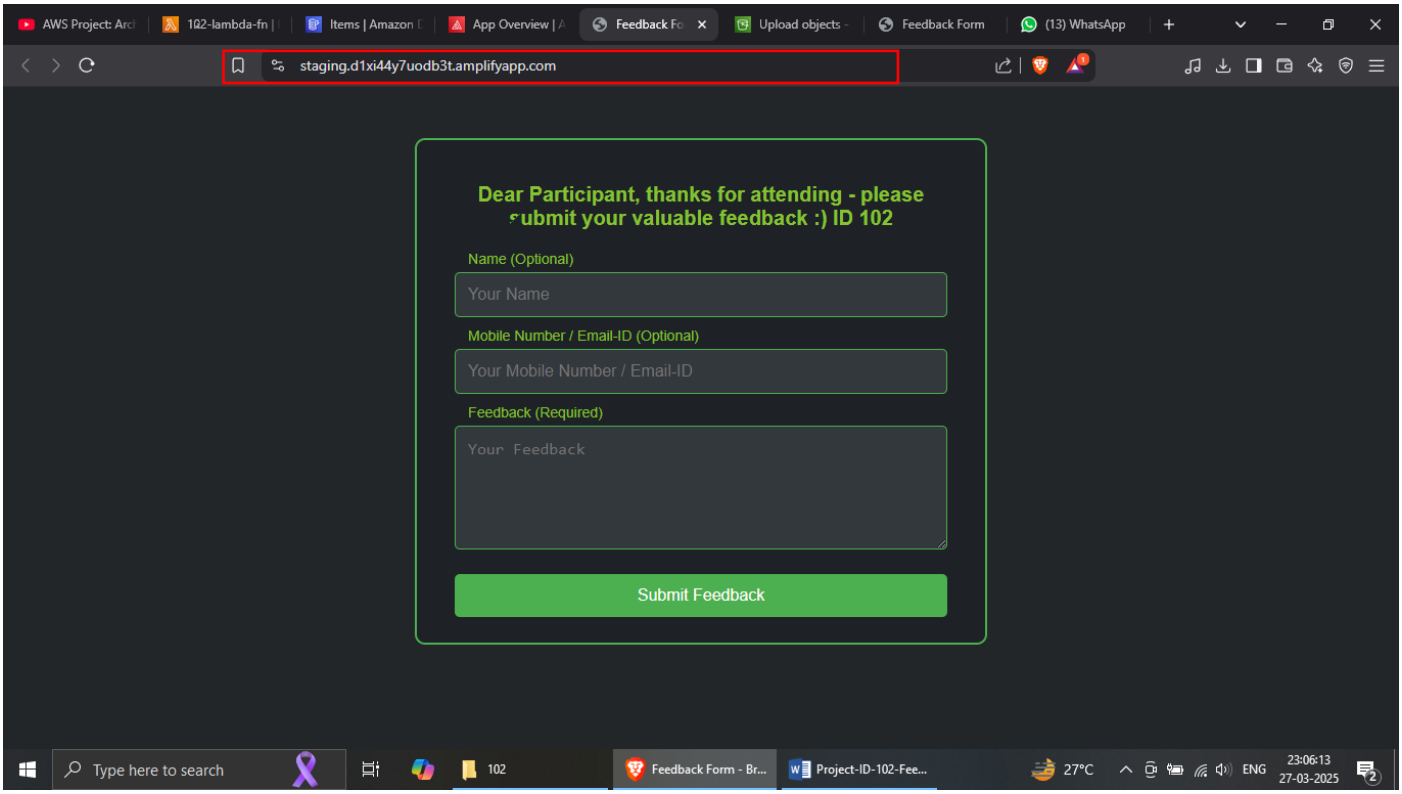


5.2 Click the domain link.

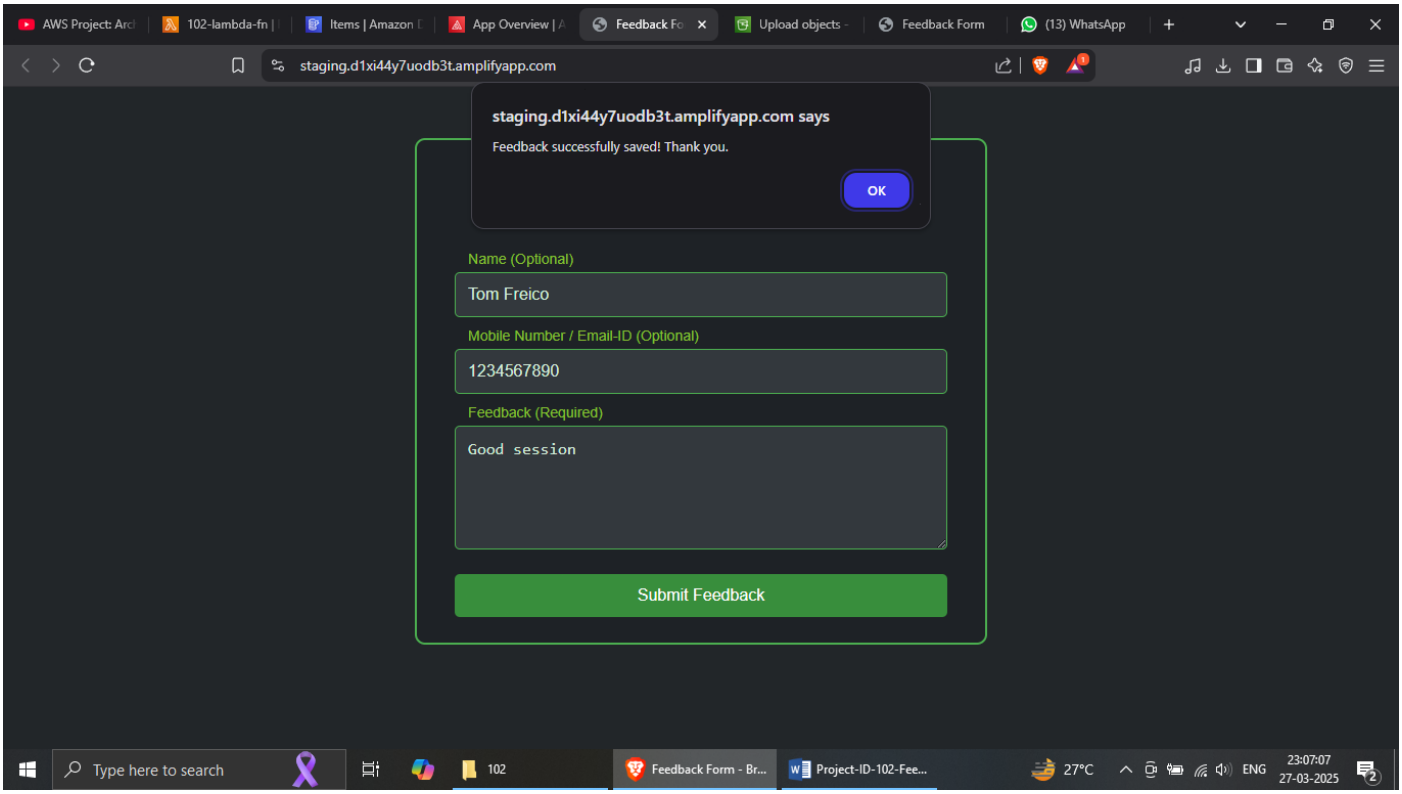


---

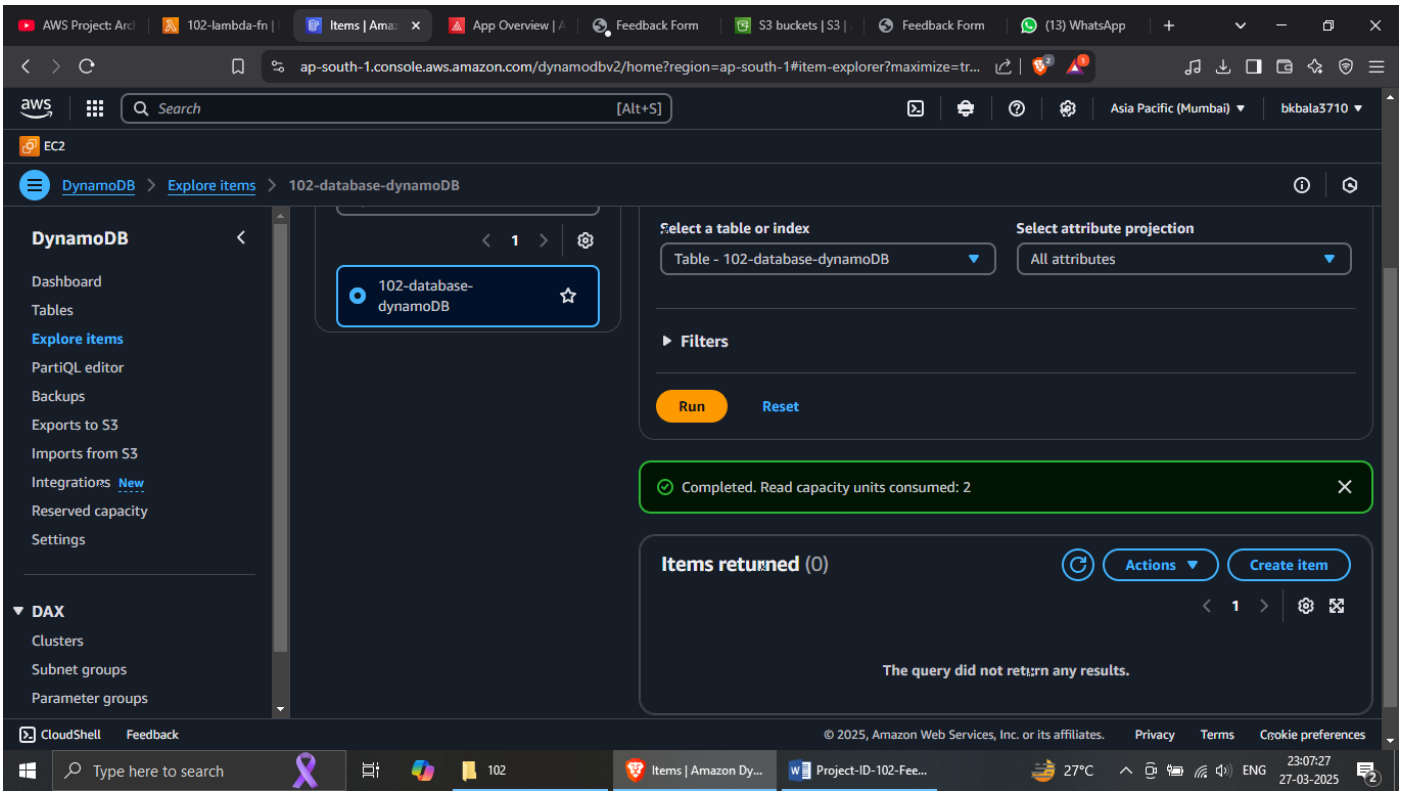
*Fill the feedback.



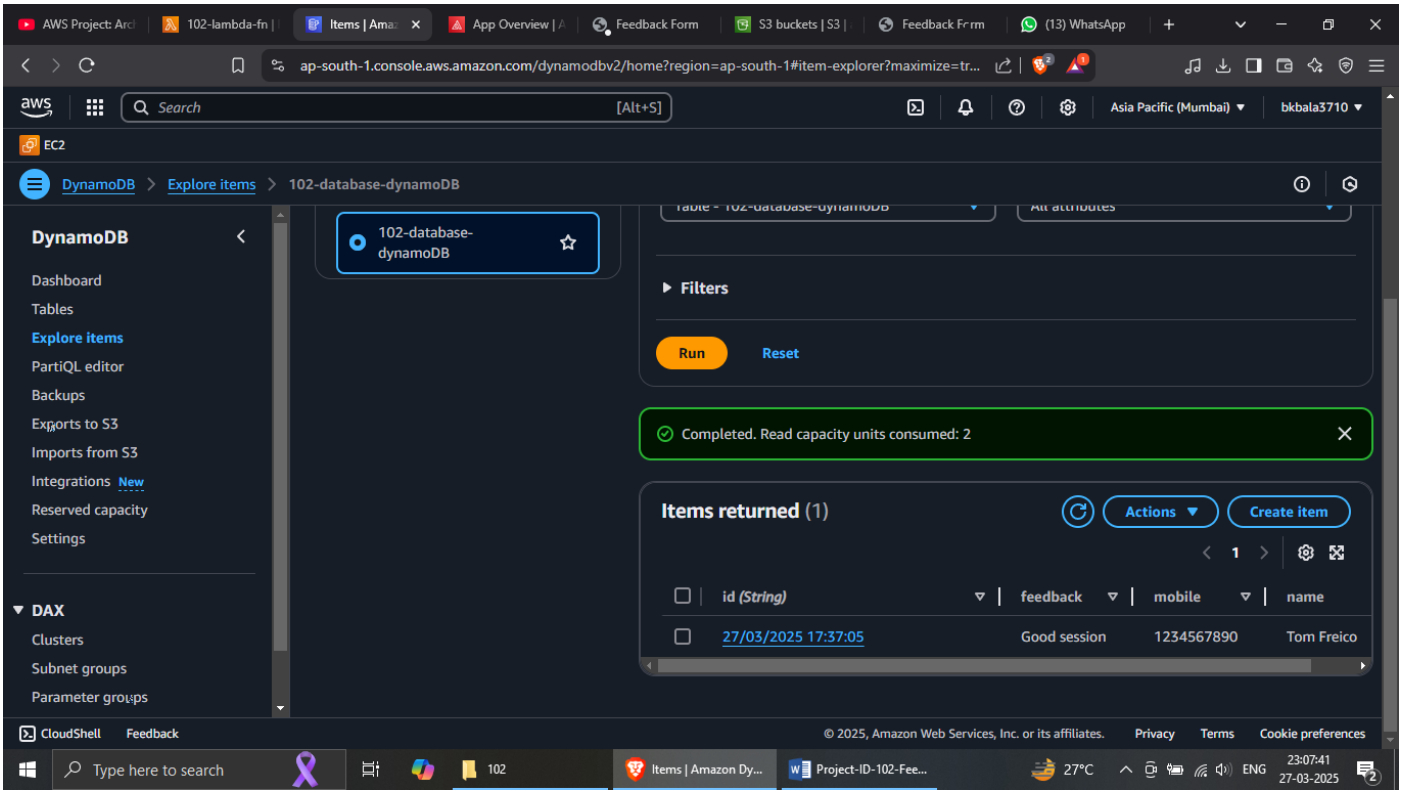
*Submitted.



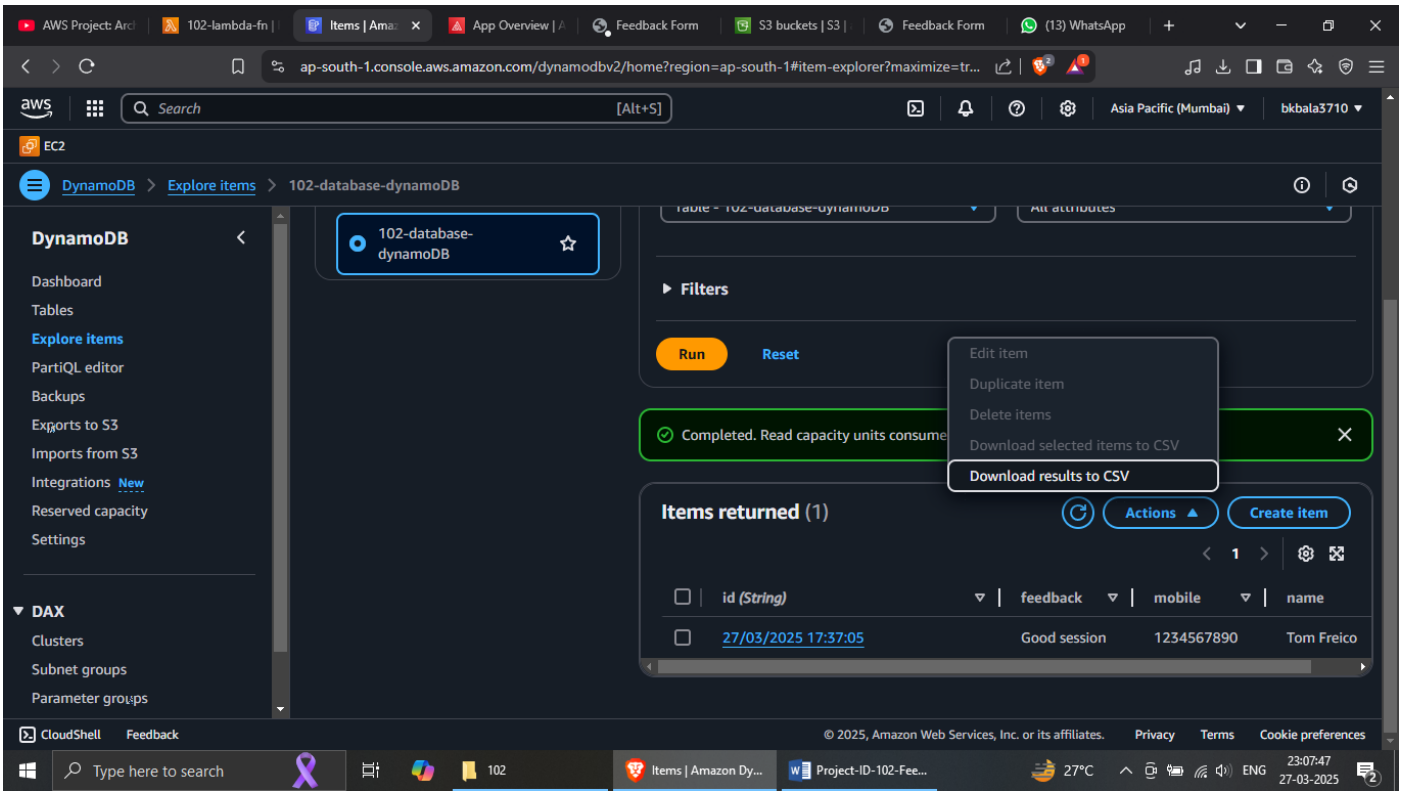
*Refresh the table.



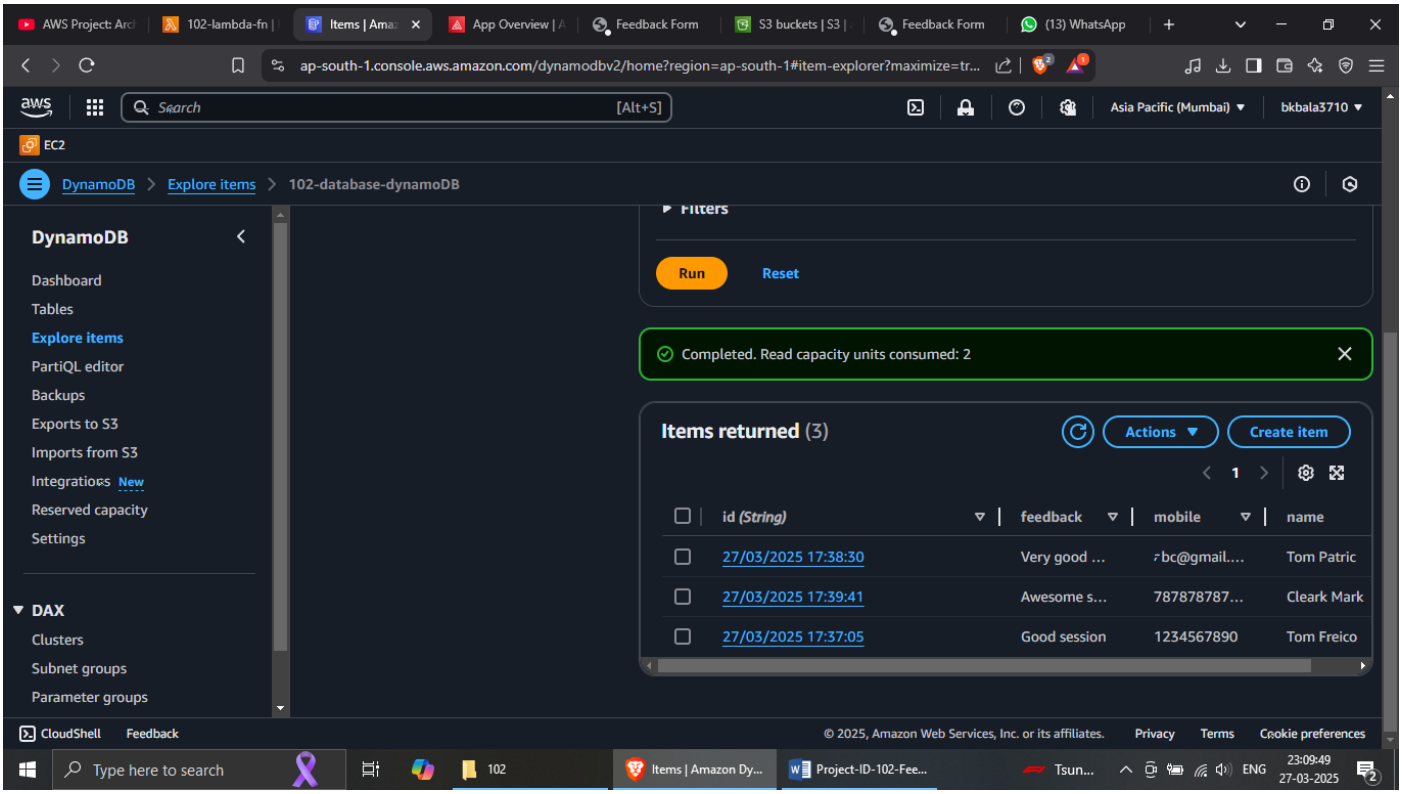
*The entries are getting placed in dynamoDB table.



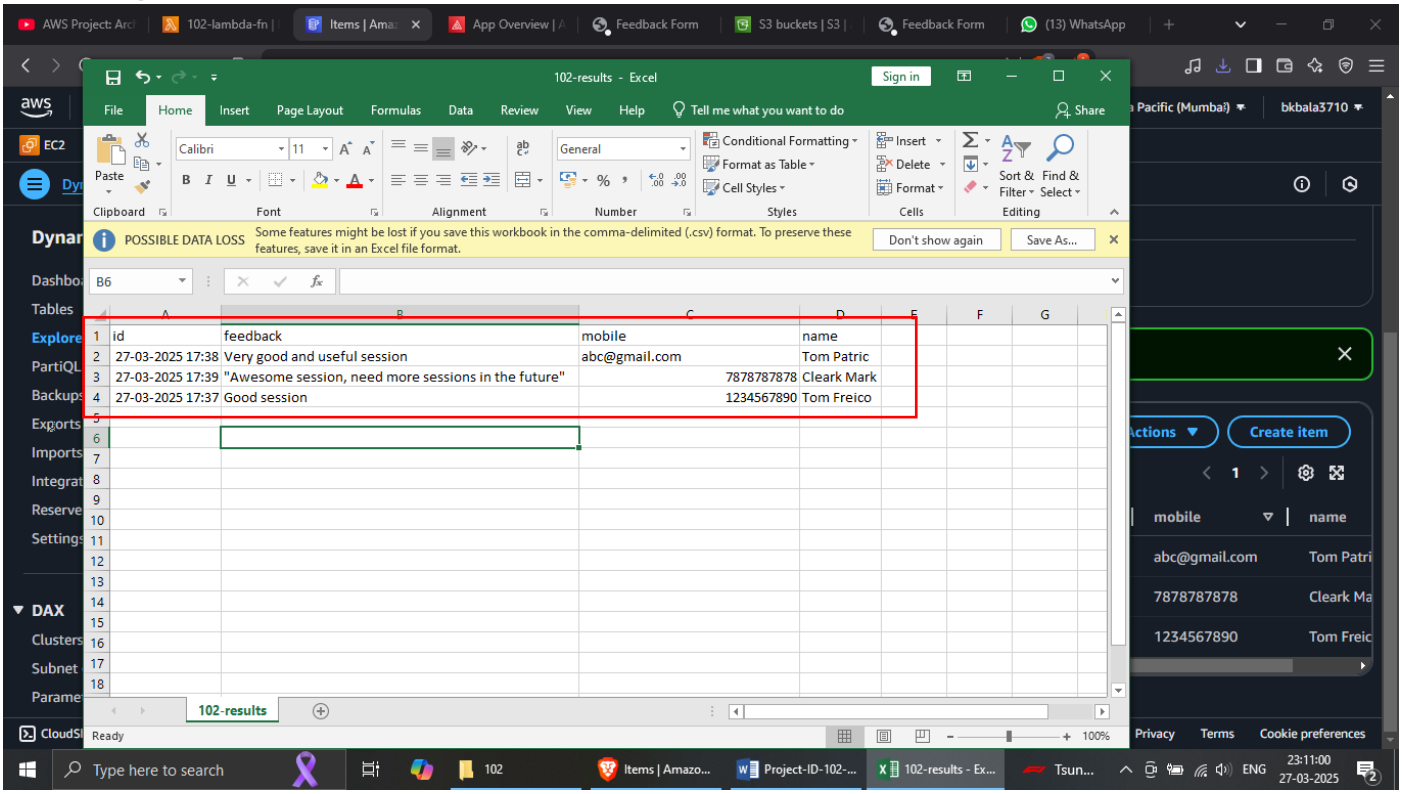
*We can download as CSV file.



*More entries...



*Its done



Reference link: https://www.youtube.com/watch?v=7m_q1ldzw0U&t=1029s (

Tiny Technical Tutorials)