

Name: Rathna Sekhar Reddy V

Emp.id: ENC/16811

Question:

Create a serverless web application that does CRUD operation on data through an HTML form hosted on EC2, processes it via Lambda functions, and stores it in DynamoDB.

STEP 1: Create DynamoDB Table

1. Open the DynamoDB console

- Go to AWS Console → DynamoDB → Tables → Create table

2. Table details

- Table name: UserSubmissions
- Partition key: submissionId (String)
- Leave Sort key empty
- Choose Provisioned or On-demand capacity (On-demand is fine for simplicity)

3. Click Create table

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with links like Dashboard, Tables, Explore items, PartQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for DAX with Clusters, Subnet groups, and Parameter groups. The main area shows a list of tables under 'Tables (1)'. One table, 'UserSubmissions', is selected and highlighted with a blue border. The right side of the screen displays the 'UserSubmissions' table details. Under the 'General information' tab, the partition key is listed as 'submissionId (String)', capacity mode as 'On-demand', and item count as 0. Other visible settings include sort key (empty), table status (Active), point-in-time recovery (PITR) (Off), table size (0 bytes), and resource-based policy (Not active). There are tabs for Actions, Indexes, Monitor, Global tables, Backups, and Exports.

STEP 2: Create IAM Role for Lambda

1. Go to IAM → Roles → Create role

- Select AWS Service
- Use case: Lambda
- Click Next

2. Attach permissions

- AmazonDynamoDBFullAccess (*for simplicity — can later restrict to the table*)

STEP 3: Create Lambda Functions

You'll create **two** functions:

◆ A. Submission Lambda (POST /submit)

1. Go to AWS Lambda → Create function
 - a. Name: SubmissionFunction
 - b. Runtime: Python 3.9
 - c. Execution role: Use existing role → LambdaDynamoDBRole
2. Click Create function

The screenshot shows the AWS Lambda console interface. At the top, the navigation bar includes 'Search [Alt+S]', account information ('Account ID: 0335-6644-3169'), and region ('Asia Pacific (Mumbai)'). The main content area is titled 'SubmissionLambda' and shows the 'Function overview' tab selected. The function details include:

- Description:** -
- Last modified:** 3 hours ago
- Function ARN:** arn:aws:lambda:ap-south-1:033566443169:function:SubmissionLambda
- Function URL:** -

A diagram on the left illustrates the function's architecture, showing it connected to an 'API Gateway' trigger. Below the diagram, there are buttons for '+ Add destination' and '+ Add trigger'. At the bottom of the main panel, tabs for 'Code', 'Test' (which is selected), 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible. The 'Tutorials' sidebar on the right provides a link to a tutorial on creating a simple web app using Lambda.

◆ B. Query Lambda (GET /submissions)

1. Create another Lambda:

- a. Name: QueryFunction
- b. Runtime: Python 3.9
- c. Role: Use existing role → LambdaDynamoDBRole

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Lambda' is selected under 'Functions'. The main page title is 'QuerySubmissionsLambda'. On the left, there's a 'Function overview' section with tabs for 'Diagram' and 'Template'. It shows a box labeled 'QuerySubmissionsLambda' with a 'Layers' section below it. To the right of the box is an 'API Gateway' icon with a 'Triggered' status. Below the box are buttons for '+ Add destination' and '+ Add trigger'. At the bottom of this section are tabs for 'Code', 'Test', 'Monitor', 'Configuration' (which is highlighted in blue), 'Aliases', and 'Versions'. On the right side, there's a detailed view of the function's configuration: 'Description' (empty), 'Last modified' (2 hours ago), 'Function ARN' (arn:aws:lambda:ap-south-1:033566443169:function:QuerySubmissionsLambda), and 'Function URL' (Info). A 'Tutorials' sidebar on the right is open, showing a 'Create a simple web app' tutorial with a brief description and a 'Start tutorial' button.

STEP 4: Create API Gateway

1. Go to Amazon API Gateway → Create API
 - a. Choose REST API → Build
 - b. Name: UserSubmissionAPI
 - c. Endpoint Type: Regional
 - d. Click Create API

This screenshot is identical to the one above, showing the AWS Lambda console for the 'QuerySubmissionsLambda' function. The configuration tab is selected, and the sidebar on the right displays the 'Create a simple web app' tutorial.

4.1 Create Resource /submit

1. Click **Actions** → **Create Resource**
 - a. Resource name: submit
 - b. Resource path: /submit
 - c. Click **Create Resource**
2. With /submit selected → Click **Create Method** → **POST**
3. Integration type: **Lambda Function**
 - a. Select your region
 - b. Lambda Function: SubmissionFunction
 - c. Save → Allow permission prompt

The screenshot shows the AWS Lambda console interface. At the top, the account ID is 0335-6644-3169 and the region is Asia Pacific (Mumbai). The function name is 'QuerySubmissionsLambda'. In the 'Function overview' section, there is a diagram showing the function connected to an 'API Gateway'. Below the diagram, there are buttons for '+ Add destination' and '+ Add trigger'. The 'Configuration' tab is currently selected. On the right side, there is a 'Tutorials' sidebar with a section titled 'Create a simple web app' which includes a list of steps: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function through its function URL'. There is also a 'Start tutorial' button.

4.2 Create Resource /submissions

1. Create another resource /submissions
2. Add a **GET** method
 - a. Integration: **Lambda Function**
 - b. Lambda: QueryFunction
 - c. Save → Allow permission prompt

API Gateway

Resources

Resource details

Methods (2)

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

4.3 Enable CORS

1. For each resource (/submit and /submissions):
 - a. Click **Actions** → **Enable CORS**
 - b. Accept defaults and confirm

4.4 Deploy the API

1. Click **Actions** → **Deploy API**
 - a. Deployment stage: **[New Stage]**
 - b. Stage name: prod
2. Click **Deploy**

Now you'll get URLs like:

POST <https://cssjz6x5ok.execute-api.ap-south-1.amazonaws.com/prod/submit>
 GET <https://cssjz6x5ok.execute-api.ap-south-1.amazonaws.com/prod/submissions>

STEP 5: Launch EC2 Instance and Host HTML

5.1 Launch EC2

1. Go to **EC2** → **Launch Instance**
 - a. Name: WebFormServer
 - b. AMI: **Amazon Linux 2**
 - c. Instance type: **t2.micro**
 - d. Key pair: (Create one if needed)
 - e. Security Group: Allow **HTTP (port 80)** and **SSH (port 22)**

2. Click Launch Instance

5.2 Connect via SSH and install apache and create html form

5.5 Test it!

1. Open browser → enter **EC2 Public IP**
 2. Fill out the form → Submit
 3. Go to **DynamoDB** → **Tables** → **UserSubmissions** → **Explore table items**
 - a. You'll see your submission!

User Submission Form

Name
Ratna sekar

Email
ratna12@gmail.com

Message
Hello

Submit

Table: UserSubmissions - Items returned (2)

[Actions ▾](#)[Create item](#)

Scan started on October 31, 2025, 14:28:06

< 1 > |

<input type="checkbox"/>	submissionId (String)	email	message	name	status
<input type="checkbox"/>	1e2a87dc-8756-4c37-9e5...	syam@test....	Hi	Syam	Ready
<input type="checkbox"/>	00cf8fcf-4dc5-4186-a015...	john@test.c...	Hello	John	Ready