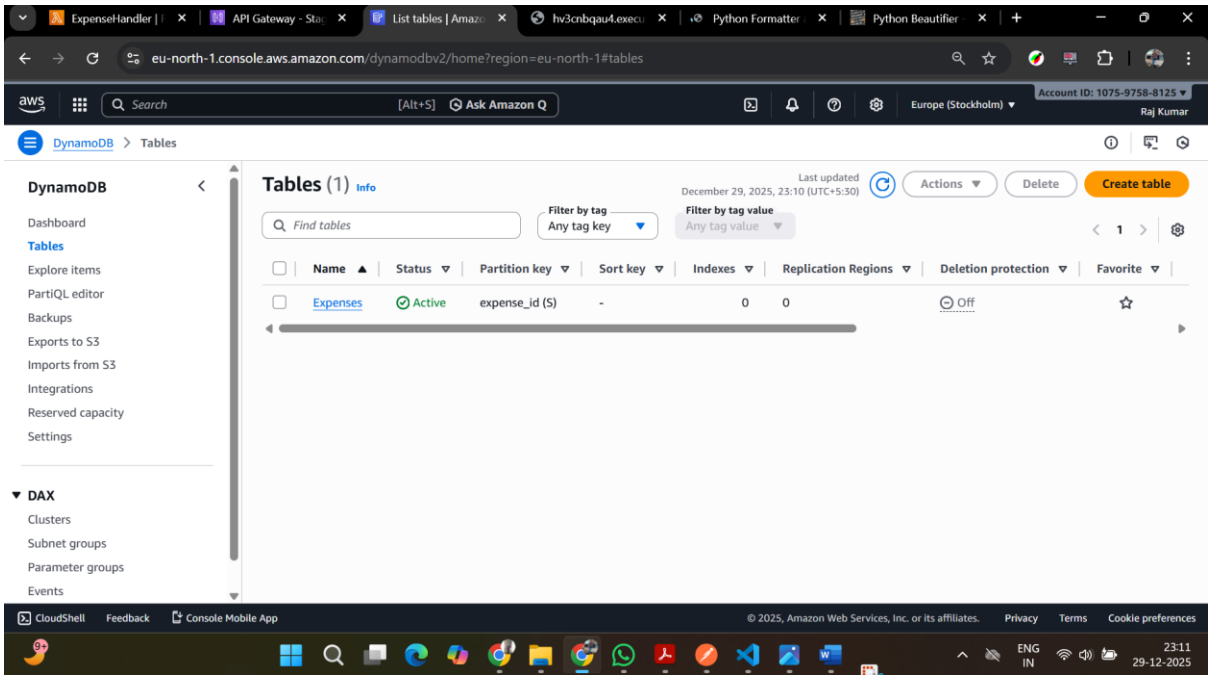
Serverless Expense Tracker – Step-by-Step Implementation

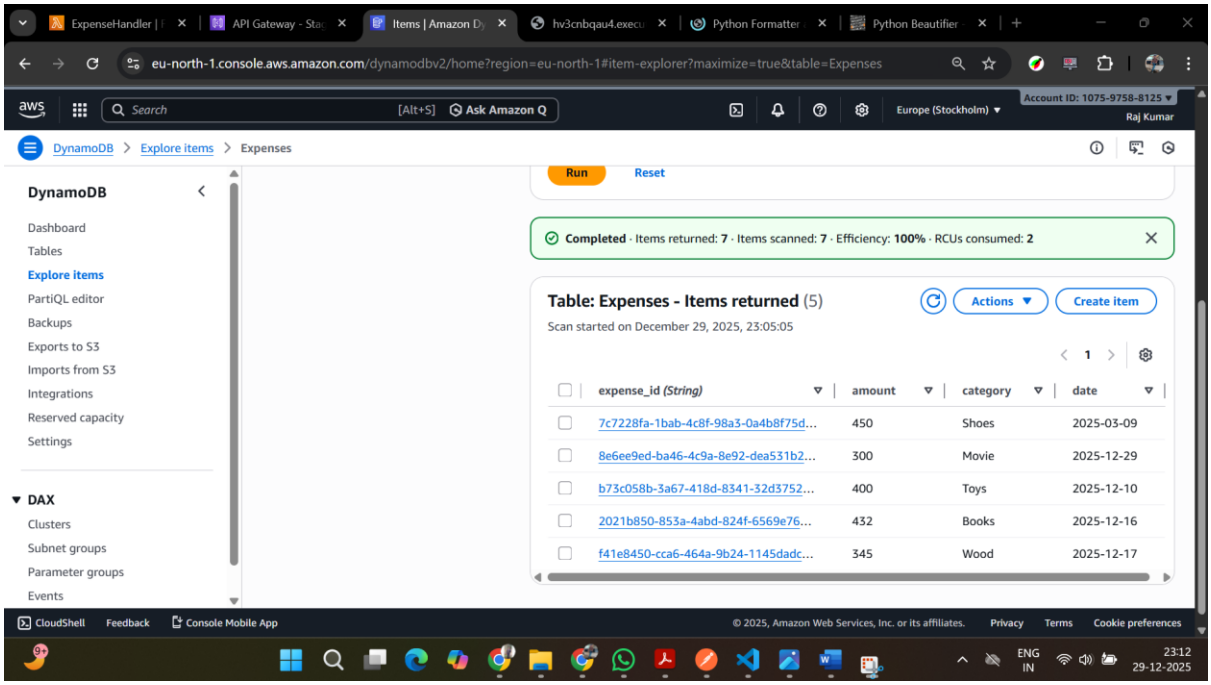
- The Serverless Expense Tracker is a cloud-based web application developed using AWS serverless services. Amazon S3 is used to host the frontend, while AWS Lambda and Amazon API Gateway handle backend processing and API management. Amazon DynamoDB is used to store expense data securely, with AWS IAM managing access permissions. The APIs were tested and validated using Postman to ensure proper request and response handling.

Step 1: Create DynamoDB Table

Logged in to the **AWS Management Console**.

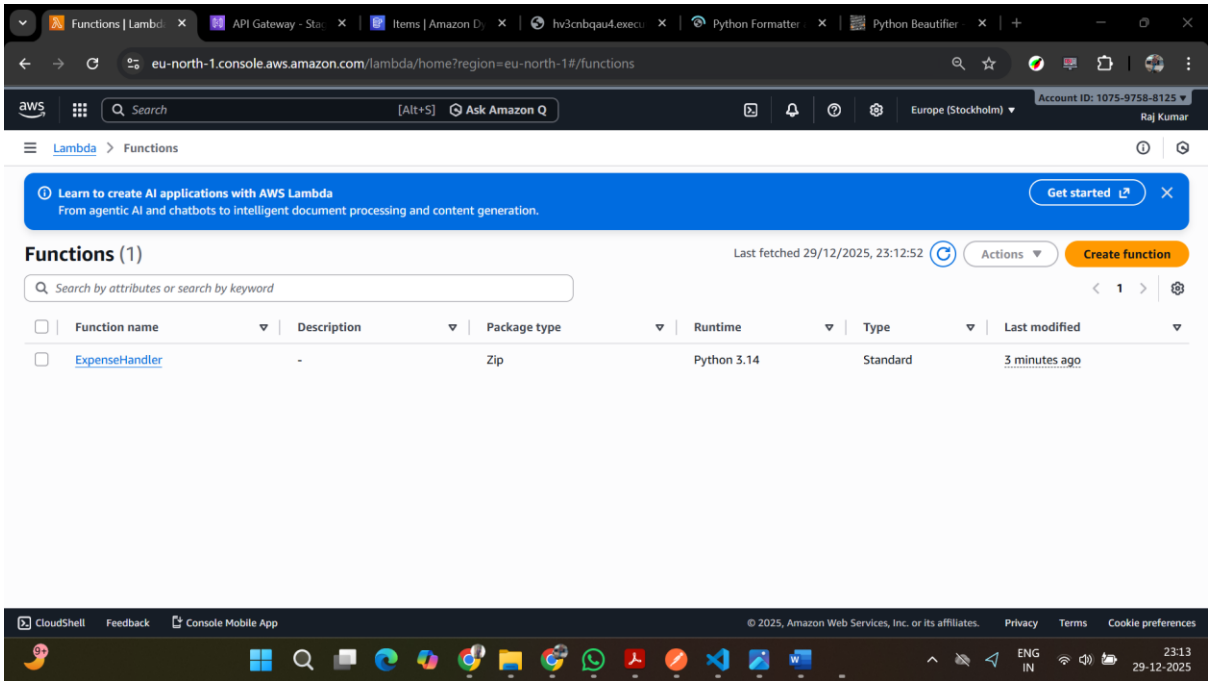
- Navigated to the **DynamoDB** service.
- Clicked on **Create table**.
- Entered the **Table name** as Expenses.
- Set the **Partition key** as:
    - Name: expense_id
    - Type: String
- Kept all other settings as default.
- Clicked **Create table**.
- Verified that the table status was **Active**.




ExpenseHandler |
API Gateway - Sta
Items | Amazon D
hv3cnbqau4.execu
Python Formatter
Python Beautifier
eu-north-1.console.aws.amazon.com/dynamodbv2/home?region=eu-north-1#item-explorer?maximize=true&table=Expenses
Search
[Alt+S]
Ask Amazon Q
Europe (Stockholm)
Account ID: 1075-9758-8125
Raj Kumar
DynamoDB
Explore items
Expenses
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Run
Reset
Completed · Items returned: 7 · Items scanned: 7 · Efficiency: 100% · RCUs consumed: 2
Table: Expenses - Items returned (5)
Scan started on December 29, 2025, 23:05:05
Actions
Create item
1
expense_id (String)
amount
category
date
7c7228fa-1bab-4c8f-98a3-0a4b8f75d...
450
Shoes
2025-03-09
8e6ee9ed-ba46-4c9a-8e92-dea531b2...
300
Movie
2025-12-29
b73c058b-3a67-418d-8341-32d3752...
400
Toys
2025-12-10
2021b850-853a-4abd-824f-6569e76...
432
Books
2025-12-16
f41e8450-cca6-464a-9b24-1145dadc...
345
Wood
2025-12-17
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ENG
IN
23:12
29-12-2025

Step 2: Create AWS Lambda Function

- Navigated to the **AWS Lambda** service.
- Clicked **Create function**.
- Selected **Author from scratch**.
- Entered the function name as ExpenseLambda.
- Selected **Runtime** as Python 3.x.
- Clicked **Create function**.



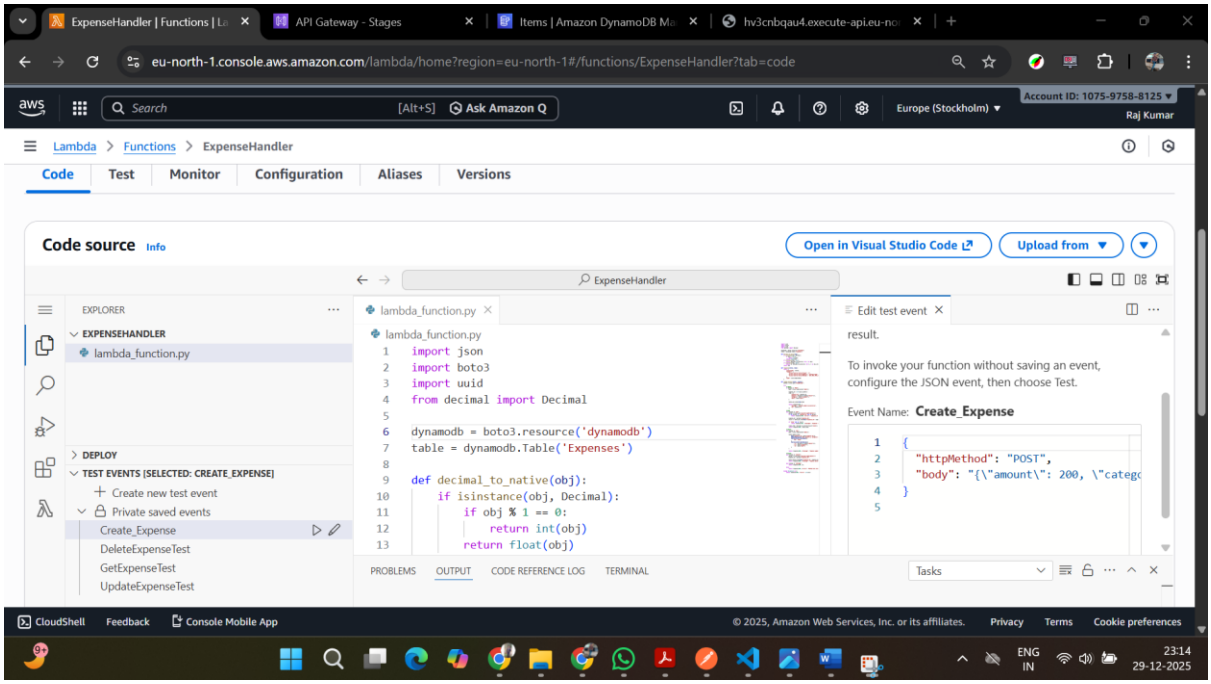
#Python Code

```
import json
import boto3
import uuid
from decimal import Decimal

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Expenses')

def decimal_to_native(obj):
    if isinstance(obj, Decimal):
```

```
        if obj % 1 == 0:
            return int(obj)
        return float(obj)
    if isinstance(obj, list):
        return [decimal_to_native(i) for i in obj]
    if isinstance(obj, dict):
        return {k: decimal_to_native(v) for k, v in obj.items()}
    return obj

def response(status, body):
    return {
        "statusCode": status,
        "headers": {
            "Access-Control-Allow-Origin": "*",
            "Access-Control-Allow-Headers": "Content-Type",
            "Access-Control-Allow-Methods": "GET,POST,PUT,DELETE,OPTIONS"
        },
        "body": json.dumps(body)
    }

def lambda_handler(event, context):
    method = event.get("httpMethod")

    try:
        # CREATE
        if method == "POST":
            data = json.loads(event["body"])

            expense_id = str(uuid.uuid4())
            item = {
                "expense_id": expense_id,
                "amount": Decimal(str(data["amount"])),
                "category": data["category"],
                "date": data["date"]
            }

            table.put_item(Item=item)

            return response(201, {
                "message": "Expense added successfully",
                "id": expense_id
```

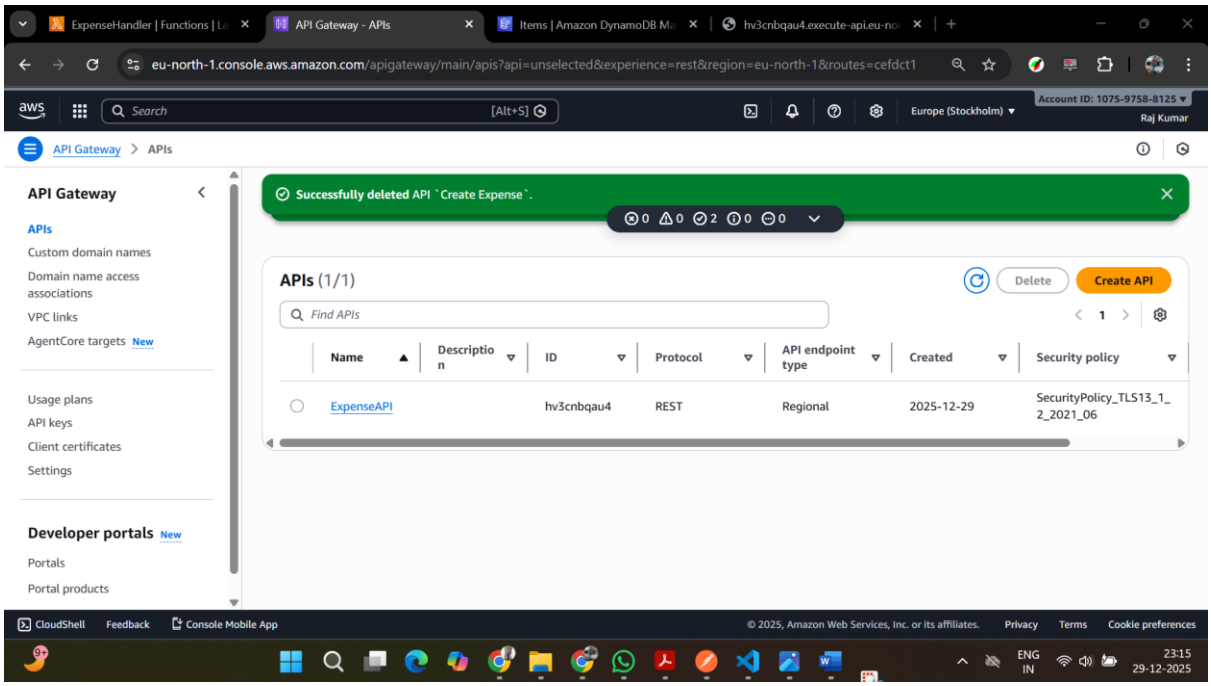
```
        })

    # READ
    elif method == "GET":
        params = event.get("queryStringParameters")
        if not params or "expense_id" not in params:
            return response(400, {"error": "expense_id is required"})

        expense_id = params["expense_id"]
        result = table.get_item(Key={"expense_id": expense_id})

        if "Item" not in result:
            return response(404, {"message": "Expense not found"})

        clean_item = decimal_to_native(result["Item"])
        return response(200, clean_item)

    # UPDATE
    elif method == "PUT":
        data = json.loads(event["body"])

        table.update_item(
            Key={"expense_id": data["expense_id"]},
            UpdateExpression="SET amount=:a, category=:c, #d=:d",
            ExpressionAttributeNames={
                "#d": "date"
            },
            ExpressionAttributeValues={
                ":a": Decimal(str(data["amount"])),
                ":c": data["category"],
                ":d": data["date"]
            }
        )

        return response(200, {"message": "Expense updated successfully"})

    # DELETE
    elif method == "DELETE":
        params = event.get("queryStringParameters")
        expense_id = params["expense_id"]
```

```
        table.delete_item(Key={"expense_id": expense_id})
        return response(200, {"message": "Expense deleted successfully"})

    elif method == "OPTIONS":
        return response(200, {})

    else:
        return response(405, {"error": "Method not allowed"})

except Exception as e:
    return response(500, {"error": str(e)})
```



Step 3: Create REST API Using API Gateway

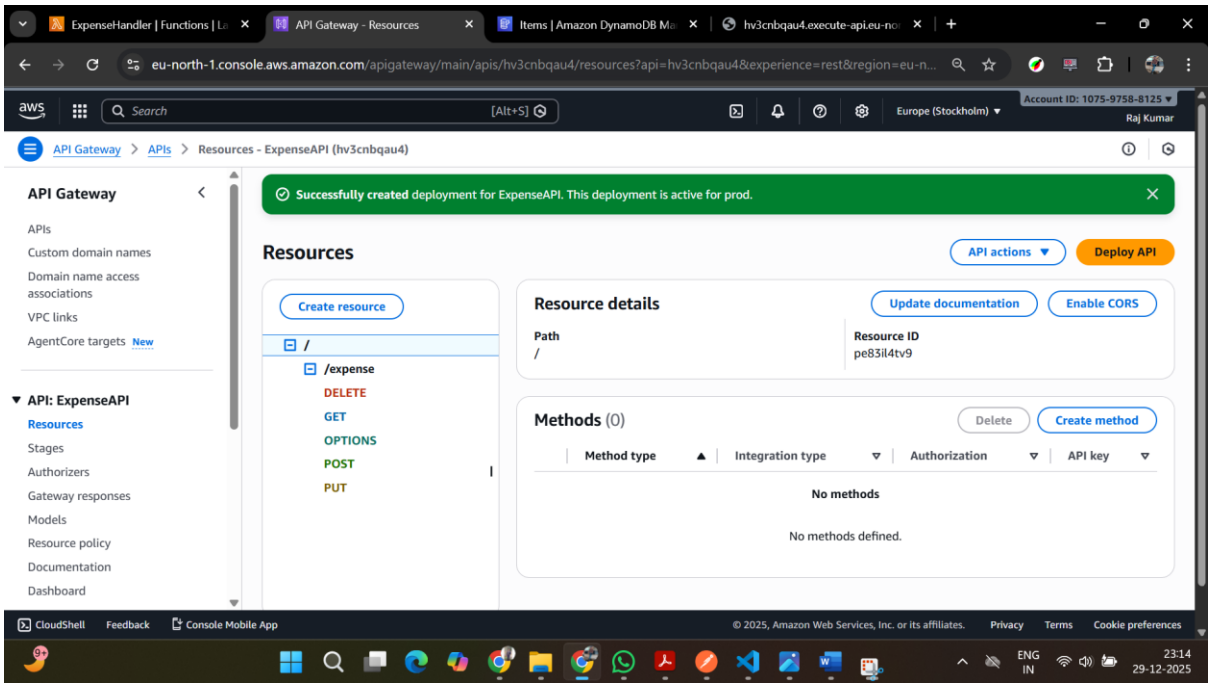
- Opened the **API Gateway** service.
- Clicked **Create API**.
- Selected **REST API** and clicked **Build**.
- Entered the API name as ExpenseAPI.



Create Resource and Methods

- Created a resource named /expense.
- Added the following HTTP methods:
  - POST
  - GET
  - PUT
  - DELETE
  - OPTIONS
- Integrated all methods with the Lambda function.

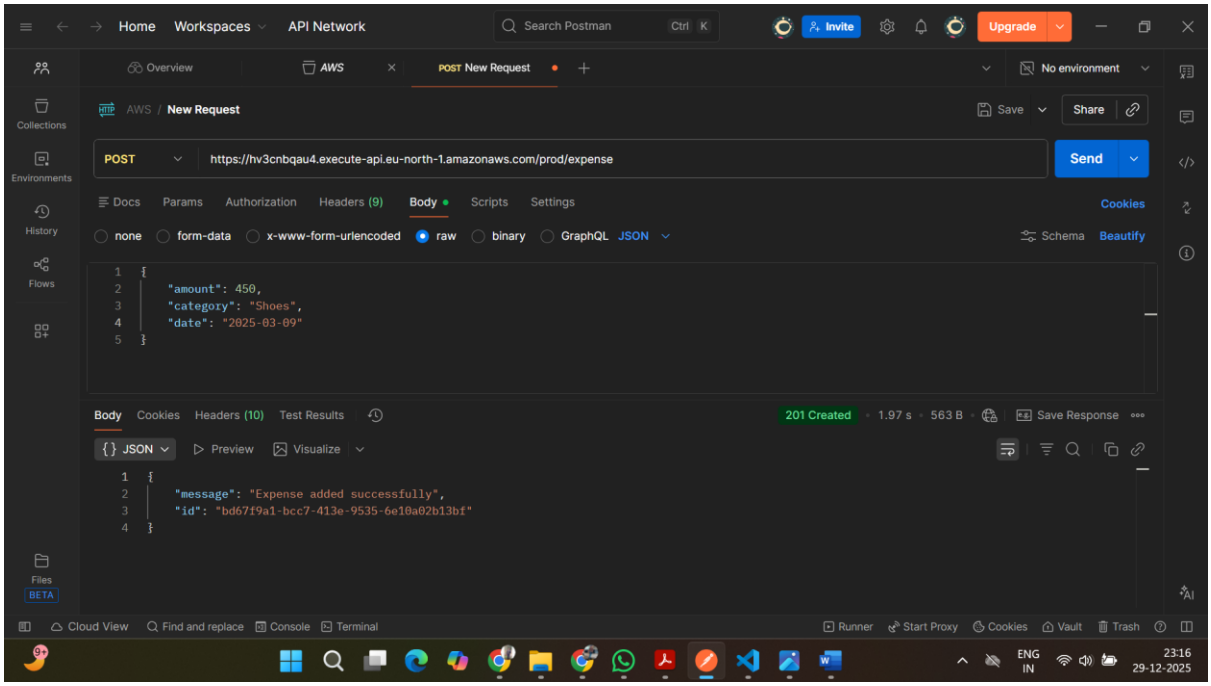
- Enabled **Lambda Proxy Integration**.
- Saved the configuration.



Enable CORS

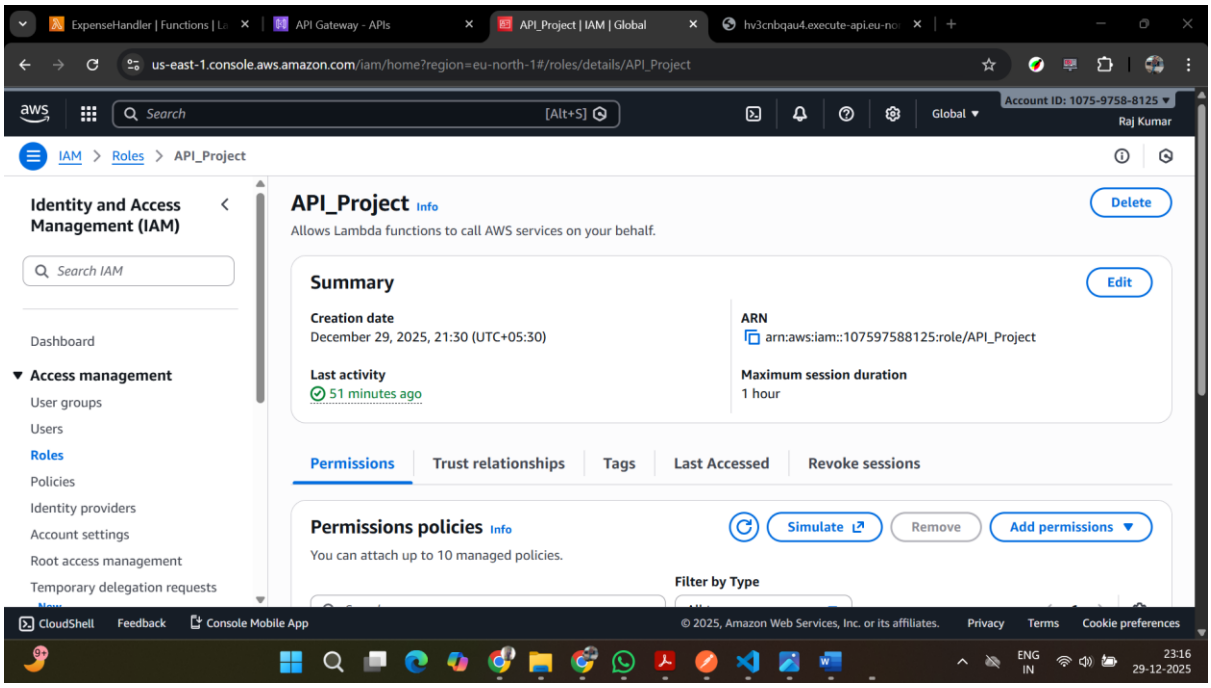
- Selected the /expense resource.
- Clicked **Enable CORS**.
- Applied CORS settings.
- Verified that CORS headers were added successfully.

Step4: API Testing Using Postman

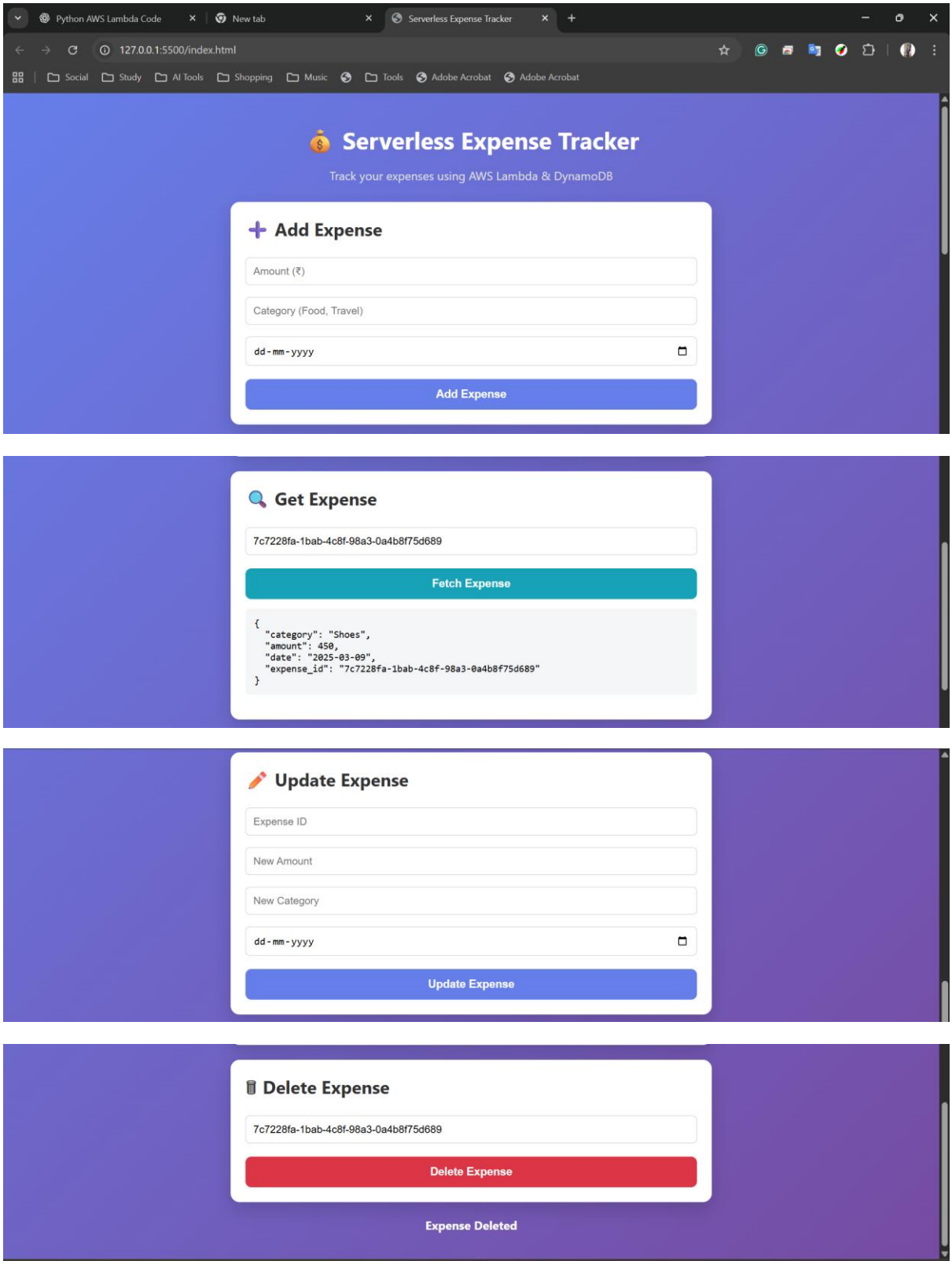


Step 5: Create IAM Role for Lambda

- Opened the **IAM** service.
- Clicked on **Roles → Create role**.
- Selected **AWS service** as the trusted entity.
- Chose **Lambda** as the use case.
    - Attached the following policies:
    - AmazonDynamoDBFullAccess
    - AWSLambdaBasicExecutionRole
- Named the role (example: LambdaExpenseRole).
- Created the role successfully.



Step 6: Created a website using HTML, CSS, JavaScript



HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Serverless Expense Tracker</title>
 <link rel="stylesheet" href="style.css" />
</head>
<body>

 <div class="container">
  <h1>💰 Serverless Expense Tracker</h1>
  <p class="subtitle">Track your expenses using AWS Lambda & DynamoDB</p>

  <div class="card">
   <h2>➕ Add Expense</h2>
   <input id="amount" type="number" placeholder="Amount (₹)" />
   <input id="category" type="text" placeholder="Category (Food, Travel)" />
   <input id="date" type="date" />
   <button onclick="addExpense()">Add Expense</button>
  </div>

  <div class="card">
   <h2>🔍 Get Expense</h2>
   <input id="getId" placeholder="Expense ID" />
   <button class="secondary" onclick="getExpense()">Fetch Expense</button>
   <pre id="getResult"></pre>
  </div>

  <div class="card">
   <h2>✏️ Update Expense</h2>
   <input id="updateId" placeholder="Expense ID" />
   <input id="updateAmount" type="number" placeholder="New Amount" />
   <input id="updateCategory" placeholder="New Category" />
   <input id="updateDate" type="date" />
   <button onclick="updateExpense()">Update Expense</button>
  </div>

  <div class="card danger">
   <h2>🗑️ Delete Expense</h2>
```

```
    <input id="deleteId" placeholder="Expense ID" />
    <button class="danger-btn" onclick="deleteExpense()">Delete Expense</button>
   </div>

   <p id="message"></p>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

CSS

```
* {
  box-sizing: border-box;
}

body {
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(135deg, #667eea, #764ba2);
  min-height: 100vh;
  margin: 0;
  padding: 20px;
}

.container {
  max-width: 650px;
  margin: auto;
}

h1 {
  text-align: center;
  color: #fff;
  margin-bottom: 5px;
}

.subtitle {
  text-align: center;
  color: #e0e0e0;
  margin-bottom: 25px;
}
```

```
.card {
  background: #ffffff;
  padding: 20px;
  margin-bottom: 20px;
  border-radius: 12px;
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
  transition: transform 0.2s ease;
}

.card:hover {
  transform: translateY(-3px);
}

.card h2 {
  margin-top: 0;
  margin-bottom: 15px;
  color: #333;
}

input {
  width: 100%;
  padding: 10px;
  margin: 8px 0;
  border-radius: 6px;
  border: 1px solid #ccc;
  font-size: 14px;
}

input:focus {
  outline: none;
  border-color: #667eea;
}

button {
  width: 100%;
  padding: 12px;
  margin-top: 10px;
  background: #667eea;
  color: #fff;
  font-size: 15px;
```

```
  font-weight: 600;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: background 0.3s ease;
}

button:hover {
  background: #5563d8;
}

.secondary {
  background: #17a2b8;
}

.secondary:hover {
  background: #138496;
}

.danger-btn {
  background: #dc3545;
}

.danger-btn:hover {
  background: #c82333;
}

pre {
  background: #f4f6f8;
  padding: 12px;
  border-radius: 6px;
  font-size: 13px;
  max-height: 200px;
  overflow: auto;
}

#message {
  text-align: center;
  font-weight: bold;
  margin-top: 15px;
  color: #fff;
```

}

JavaScript

```
const API_URL = "https://hv3cnbqau4.execute-api.eu-north-
1.amazonaws.com/prod/expense";

function showMessage(msg) {
 document.getElementById("message").innerText = msg;
}

/* ADD EXPENSE */
async function addExpense() {
 const data = {
  amount: Number(document.getElementById("amount").value),
  category: document.getElementById("category").value,
  date: document.getElementById("date").value
 };

 const res = await fetch(API_URL, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(data)
 });

 const result = await res.json();
 showMessage("Expense Added. ID: " + result.id);
 console.log(result.id);
}

/* GET EXPENSE */
async function getExpense() {
 const id = document.getElementById("getId").value;

 const res = await fetch(` ${API_URL}?expense_id=${id}` );
 const data = await res.json();

 document.getElementById("getResult").innerText =
  JSON.stringify(data, null, 2);
}
```

```
/* UPDATE EXPENSE */
async function updateExpense() {
  const expenseId = document.getElementById("updateId").value;

  const data = { expense_id: expenseId };

  const amount= document.getElementById("updateAmount").value;
  const category= document.getElementById("updateCategory").value;
  const date= document.getElementById("updateDate").value;

  if (amount) data.amount = Number(amount);
  if (category) data.category = category;
  if (date) data.date = date;

  await fetch(API_URL, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data)
  });

  showMessage("Expense Updated Successfully");
}

/* DELETE EXPENSE */
async function deleteExpense() {
  const id = document.getElementById("deleteId").value;

  await fetch(` ${API_URL}?expense_id=${id}` , {
    method: "DELETE"
  });

  showMessage("Expense Deleted");
}
```

Step 7: Uploaded the webpage in S3 bucket

- Go to S3 → Create bucket
- Bucket name: **webapp-project10**
- Region: same as EC2
- Enable:
- Versioning
- Block public access
- Create bucket

Step 8: Upload Static Content to S3

- Open the S3 bucket
- Upload:
    - index.html
    - style.css
    - script.js