**Setting Up DynamoDB Table for Recording Website Visits**

First, we will create a DynamoDB table to record all visits to our newly created website. We will use timestamp as our partition key and page (web page) as our sort key, leaving everything else as the default.A screenshot of a computer

Description automatically generated

**Creating the Lambda Function**

Next, we will create a Lambda function that captures and records events along with timestamps whenever visitors access our website. These recorded events will be sent to our DynamoDB table for data analysis. We will create this from scratch using Python 3.12 and allow the execution role to be created with basic Lambda permissions.A screenshot of a computer

Description automatically generated

If using basic permissions, you will have to add the following policy under IAM role afterwards:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "dynamodb:PutItem",

"Resource": "arn:aws:dynamodb:us-west-2:758719472525:table/vitalitySphere-DB"

}

]

}

A screenshot of a computer

Description automatically generated

Our lambda function will access our DynamoDB table and insert events as they occur in real time, we will harness the boto3 library to gain access to our database table.

A screenshot of a computer program

Description automatically generated

**Setting Up the API Gateway**

We will then set up an API Gateway to trigger our Lambda function. Once the API is created, we will edit it as needed.A screenshot of a computer

Description automatically generated

We will simply create and edit upon completion of API.

A screenshot of a computer

Description automatically generated

Next, we will add all the CORS settings. We will add the following headers:

Content-type: Specifies the media type of the resource. For example, application/json indicates that the content is in JSON format. This is important for the API to correctly parse the incoming request body.

x-amz-date: The date and time, in ISO 8601 format, when the request was signed. This is used to ensure the request is within a certain time frame and hasn't expired. For example, x-amz-date: 20230718T000000Z.

authorization: Contains the information required for the client to authenticate with the server. For AWS, this includes the AWS access key, the signed request, and the region. An example format is Authorization: AWS4-HMAC-SHA256 Credential=<ACCESS\_KEY>/20230718/us-west-2/execute-api/aws4\_request, SignedHeaders=content-type;host;x-amz-date, Signature=<SIGNATURE>.

x-api-key: A custom header used to pass the API key to the API Gateway. This is used for API key-based authentication and access control. For example, x-api-key: your-api-key.

And we will allow GET and POST methods.

A screenshot of a computer

Description automatically generated

Next we will add integration with our Lambda function

A screenshot of a computer

Description automatically generated

We can now finally deploy our API.

A screenshot of a computer

Description automatically generated

Once deployed we will be given an invoke URL, this URL will allow us to POST the timestamp and page events to our DynamoDB table.

A screenshot of a computer

Description automatically generated

**Testing the API**

To test the API, we will use the following curl command with our API's URL:curl -X POST \

https://lj41ar0ulj.execute-api.us-west-2.amazonaws.com/recordVisit \

-H 'Content-Type: application/json' \

-d '{

"page": "https://example.com/page"

}'

A screen shot of a computer code

Description automatically generated

Troubleshooting Policy Issues

For troubleshooting purposes, we needed to modify the IAM policy to allow the Lambda function to write to the DynamoDB table. The following policy was added:{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "dynamodb:PutItem",

"Resource": "arn:aws:dynamodb:us-west-2:758719472525:table/vitalitySphere-DB"

}

]

}

Viewing Recorded Data

After updating the policy, we can view our timestamp and page data. This will be beneficial as we will add this invoke URL for the API into a JavaScript function to integrate with our website in the later part of this project.A screenshot of a computer

Description automatically generated