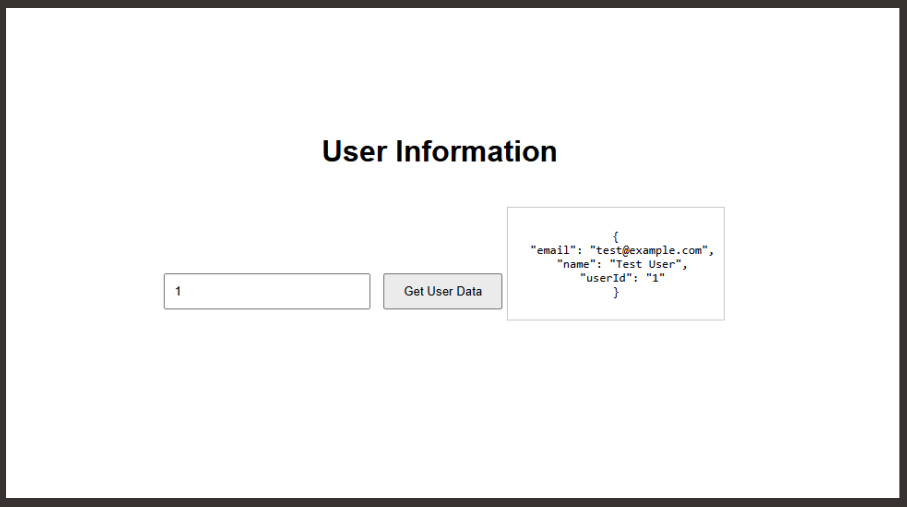
nextwork.org

Build a Three-Tier Web App

ME Melvin J Bonner



ME
Melvin J Bonner
NextWork Student
nextwork.org

Introducing Today's Project!

In this project, I will demonstrate hoe to build a three-tier web app. I'm doing this project to learn how to build a scalable web app with S3, CloudFront, Lambda, API Gateway, and DynamoDB.

Tools and concepts

Services I used were Lambda, API Gateway, and DynamoDB. Key concepts I learned include Lambda functions, DynamoDB tables, and CloudFront invalidation.

Project reflection

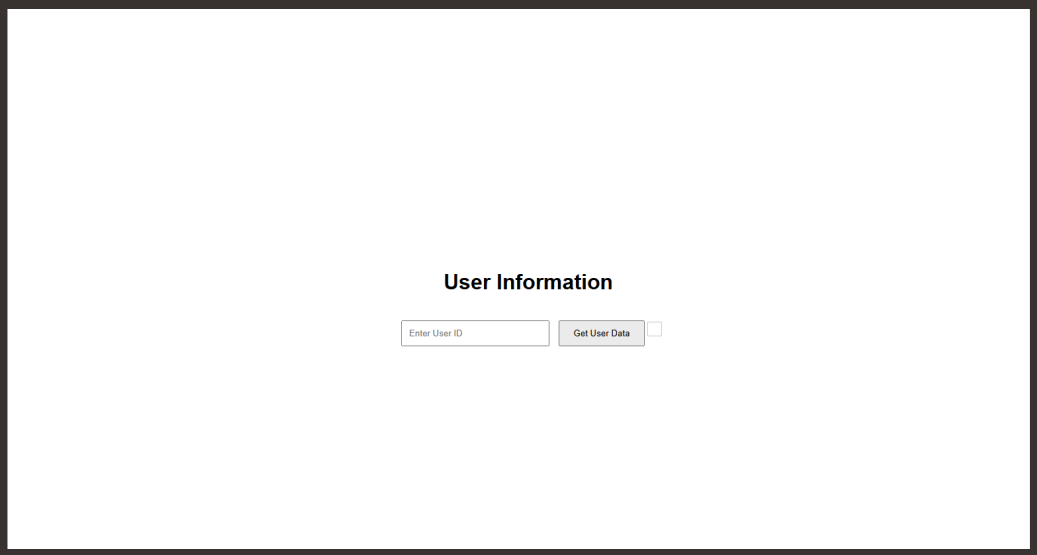
This project took me approximately 3 hours. The most challending part was getting the final web app to work. I had to invalidate the script.js file because CloudFront was using an older version of the file. The most rewarding part was fixing the script.js issue and getting the web app to work.

I did this project to be challenged, I completed the first three parts of the project and this brought everything togather. This project met my goals. It was definitely challenging.

Presentation tier

For the presentation tier, I will set up a S3 bucket. I will set up the S3 bucket because it will will store my website's files.

I accessed my delivered website by accessing the URL that CloudFront will use to serve my website.



Logic tier

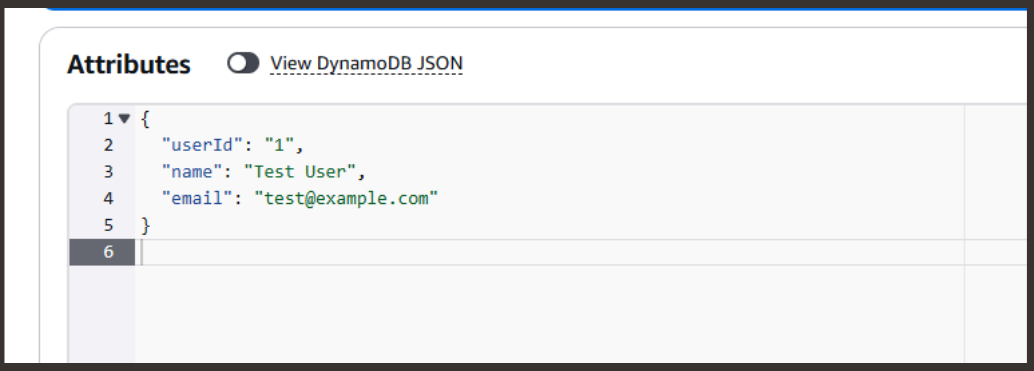
For the logic tier, I will set up an API Gateway and a Lambda Function because the API Gateway will handle requests and the Lamba Function will retreive the user data from the DynamoDB table.

The Lambda function retrieves data by being triggered by events, once triggered, the Lambda service executes your code in the runtime environment you've picked.

Data tier

For the data tier, I will set up a DynamoDB because I will need it to store user data.

The partition key for my DynamoDB table is like a lable which means that you can use it to group similar items.



ME
Melvin J Bonner
NextWork Student

Logic and Data tier

Once all three layers of my three-tier architecture are set up, the next step is to integrate the tiers of my application.

To test my API, I took the invoke URL from the API Gateway console and added /users?userId=1 to the end of the URL. The results verified the logic and data tier's intergration.

ME
Melvin J Bonner
NextWork Student

Console Errors

The error in my distributed site was because the script.js file didn't have the updated API.

To resolve the error, I updated script.js by adding the updated API information to the file. I then reuploaded it into S3 to replace the outdated script.js file.

I ran into a second error after updating script.js. This was an error with the CORS setting in API Gateway because CORS isn't configured, it is blocking the request from CloudFront.


Welcome
Elements
Console
Sources
Network
top
Filter
Default levels
11
GET https://d2rqw1nzmrr6gh.cloudfront.net/favicon.ico 403 (Forbidden)
favicon.ico:1
[NEW] Explain Console errors by using Copilot in Edge: click to explain an error. Learn more
Don't show again
GET https://d2rqw1nzmrr6gh.cloudfront.net/[YOUR-PROD-API-URL]/users?userId=1 403 (Forbidden)
script.js:9
fetchUserData @ script.js:9
onclick @ (index):13
Failed to fetch user data: SyntaxError: Unexpected token '<', "<?xml vers"... is not valid JSON
script.js:19
fetchUserData @ script.js:19
await in fetchUserData
onclick @ (index):13

Resolving CORS Errors

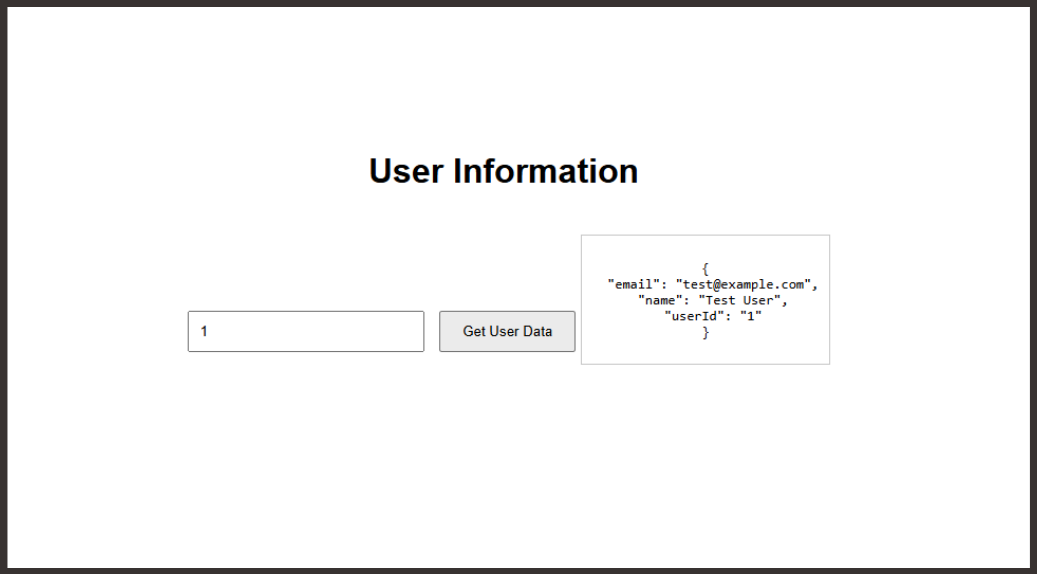
To resolve the CORS error, I first selected Enable CORS then selected GET and Options under Access Control Allow Methods.

I also updated my Lambda function because I needed to add the CORS header into the Lamba function. The changes I made was updating the * for Access-Control Allow-Origin to my CloudFront domain.

```
// Import individual components from the DynamoDB client package
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";

const ddbClient = new DynamoDBClient({ region: 'us-east-1' });
const ddb = DynamoDBDocumentClient.from(ddbClient);

async function handler(event) {
    const userId = event.queryStringParameters.userId;
    const params = {
        TableName: 'UserData',
        Key: { userId }
    };

    try {
        const command = new GetCommand(params);
        const { Item } = await ddb.send(command);

        if (Item) {
            return {
                statusCode: 200,
                headers: {
                    'Content-Type': 'application/json',
                    'Access-Control-Allow-Origin': 'https://d2rqw1nzmrr6gh.cloudfront.net/' // Allow CORS for all origins, replace
                },
                body: JSON.stringify(Item)
            };
        } else {
            return {
                statusCode: 404,
```

Fixed Solution

I verified the fixed connection between API Gateway and CloudFront by going back to my CloudFront site and invoking the user data.



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects