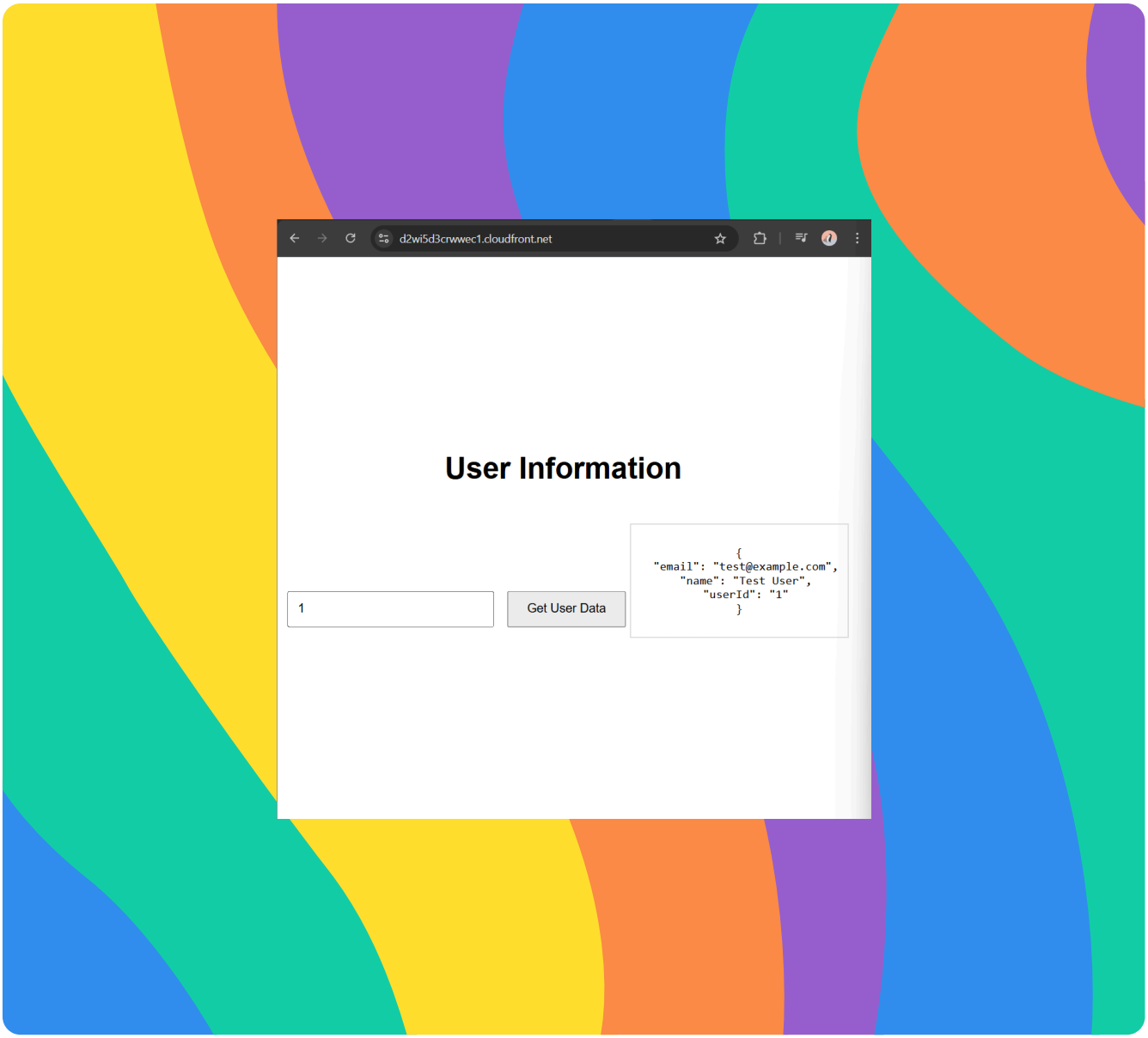
nextwork.org

Build a Three-Tier Web App

jyothesh karnam



jyothesh karnam
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to setup a three-tier web app from scratch! I will start with the presentation tier, then set up the logic tier and finally set up the data tier before tying them all together.

Tools and concepts

Services I used were Amazon S3, CloudFront, DynamoDB, Lambda, and API Gateway. Key concepts I learnt include Lambda functions, CORS errors, updating the JavaScript file with the API Invoke URL, and testing the Invoke URL within the browser.

Project reflection

This project took me approximately 1.5 hours including demo time. The most challenging part was resolving the CORS errors in both API Gateway and Lambda. It was most rewarding to see the final outcome – data getting returned in my web app!

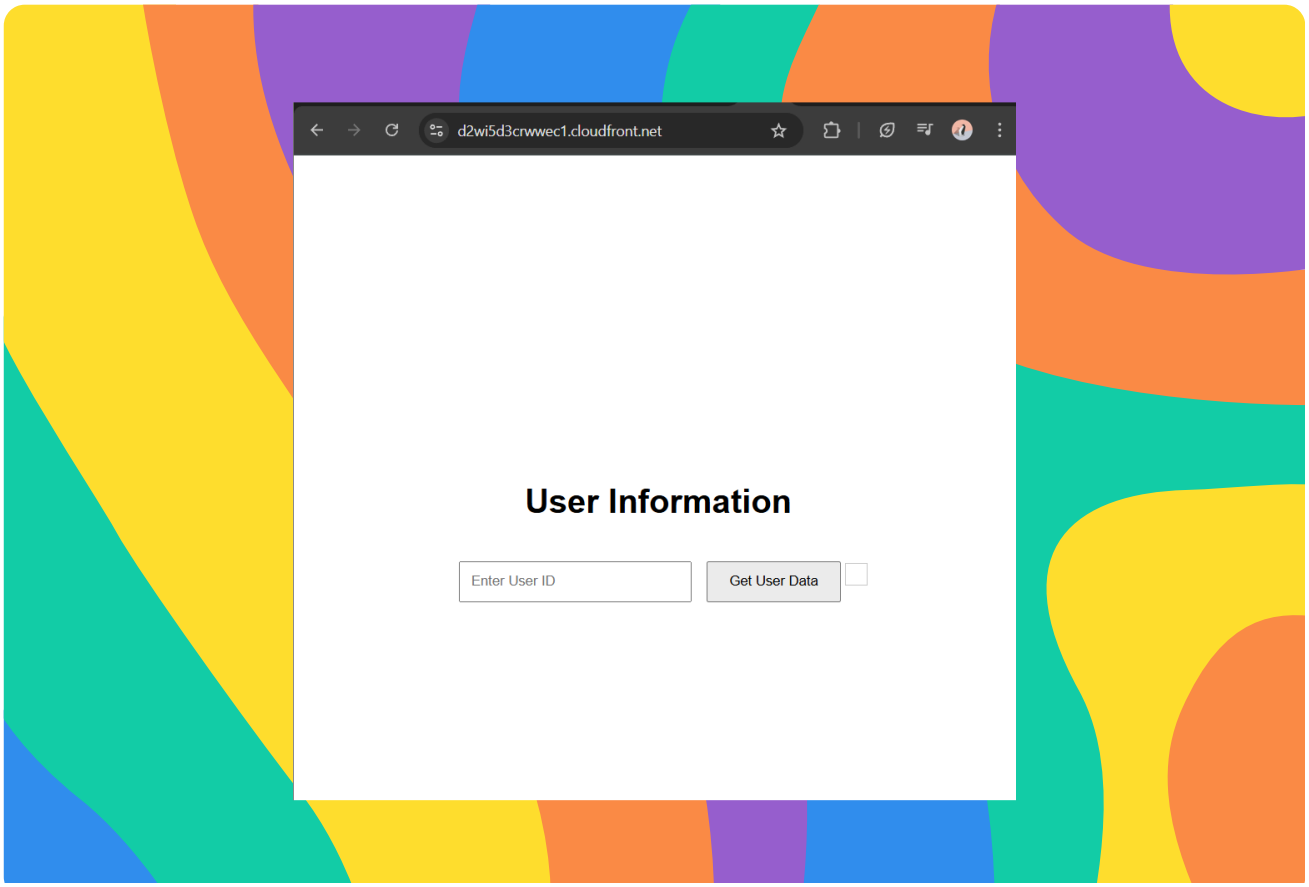
I did this project today to learn about the three-tier architecture and set up my own web app. This project absolutely met my goals, and I was able to see a fully functioning web app by the end.

jyothesh karnam
NextWork Student

Presentation tier

For the presentation tier, I will set up how my website will be displayed and available to my end users. This is because the presentation tier covers my websites files and my website distribution. Presentation tier is responsible for storing my websites files (Amazon S3) and website distribution (Amazon CloudFront).

I accessed my delivered website by visiting the CloudFront distribution's URL. This URL is working because I also set up an origin access control that lets my S3 bucket restrict access to only my CloudFront distribution.

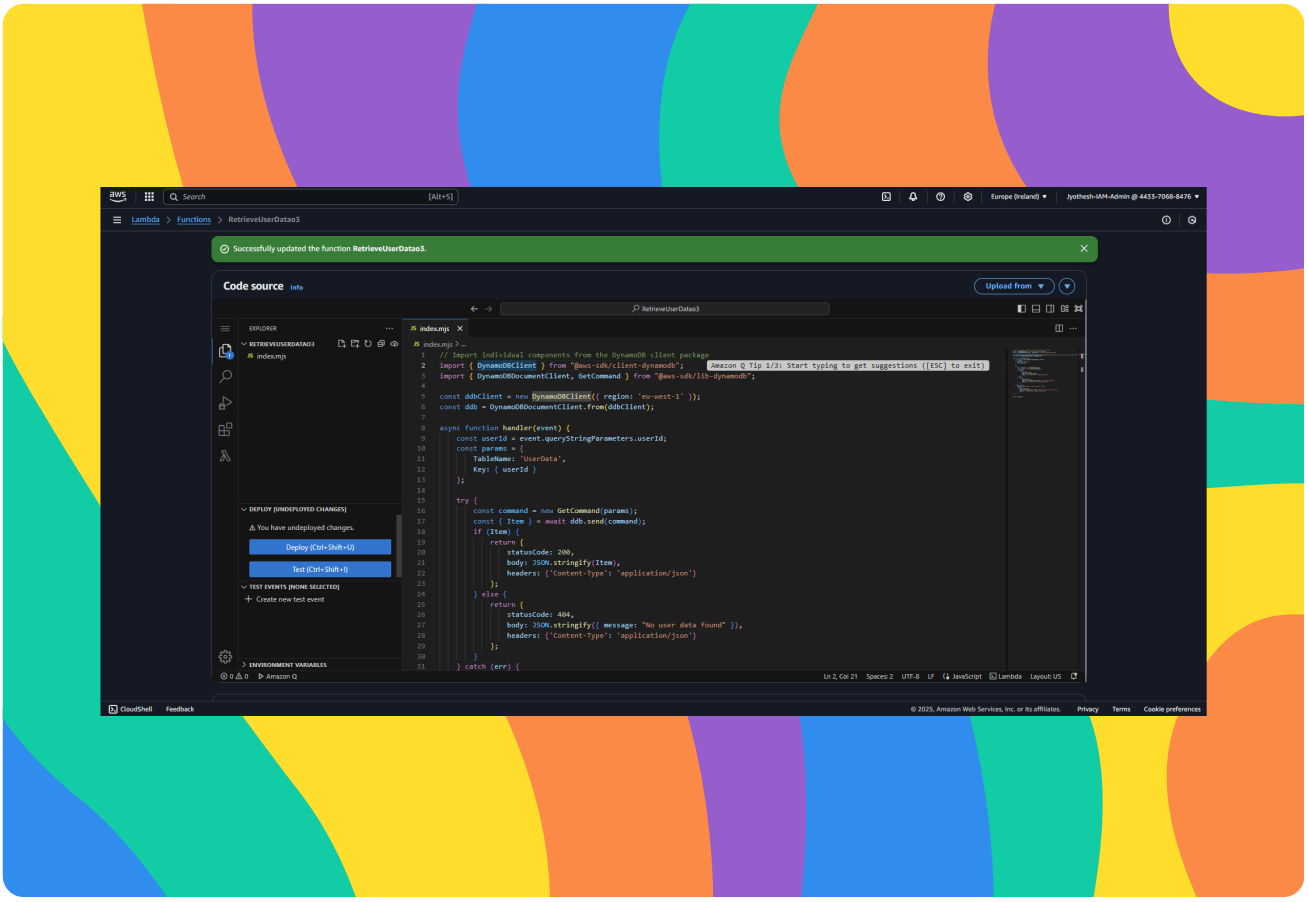


jyothesh karnam
NextWork Student

Logic tier

For the logic tier, I will set up a Lambda function to handle requests (e.g., look up userId to return some user data) and also an API with API Gateway to receive requests from the user and hand it over to Lambda.

My Lambda function retrieves data by looking up a userId (that the user enters over the web app) in DynamoDB. The Amazon SDK is used in the function code so we can use templates and libraries that lets us find the correct DynamoDB table + request data.

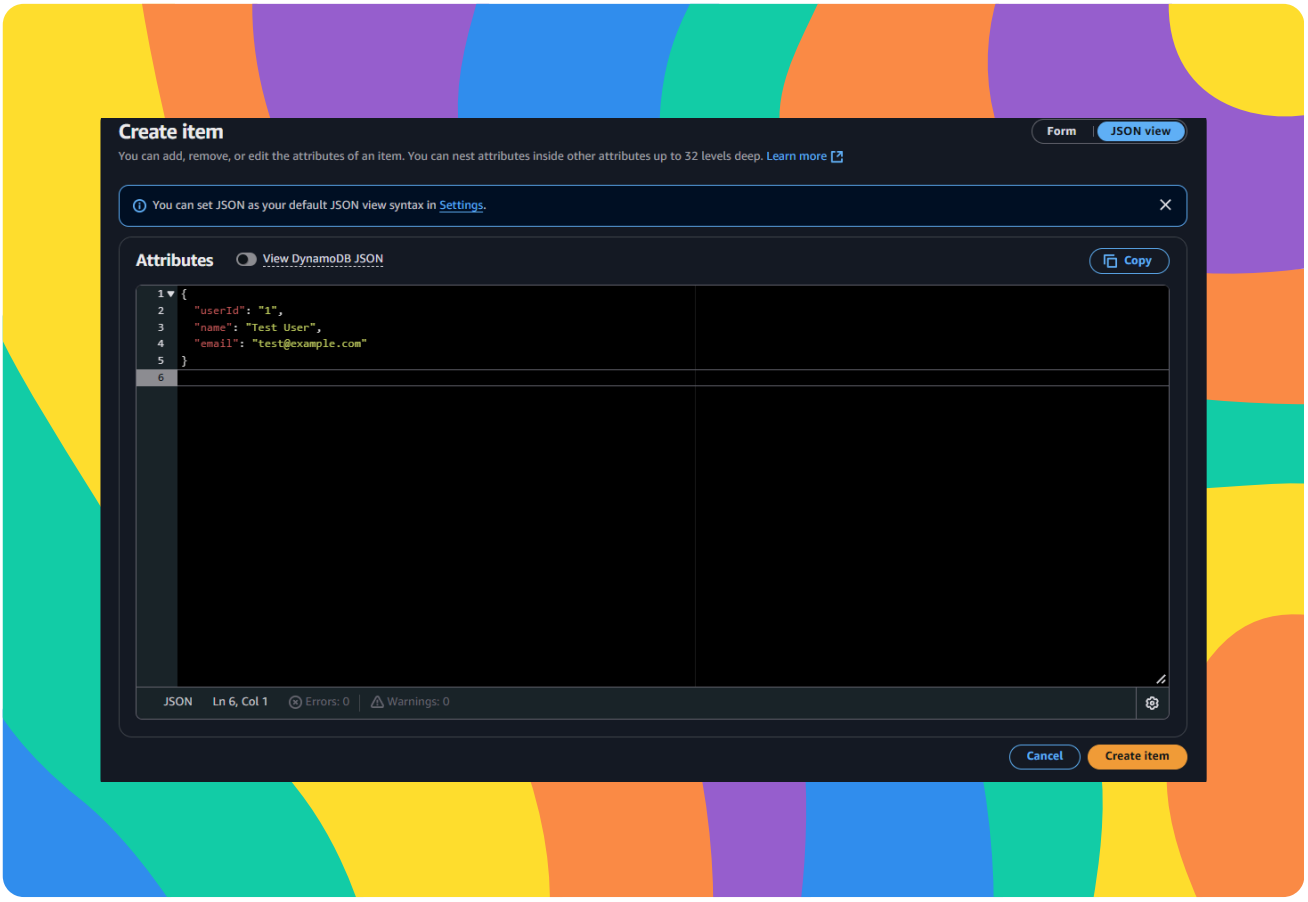


jyothesh karnam
NextWork Student

Data tier

For the data tier, I will set up a DynamoDB database that stores user data. At the moment, there is no user data for me to return to my web app's users. The data in my database will get returned once I set up DynamoDB and connect it with Lambda.

The partition key for my DynamoDB table is userId. This means that when my table looks up user data, it will look it up based on userId. Then, it can return all data (values) related to the item with that ID.

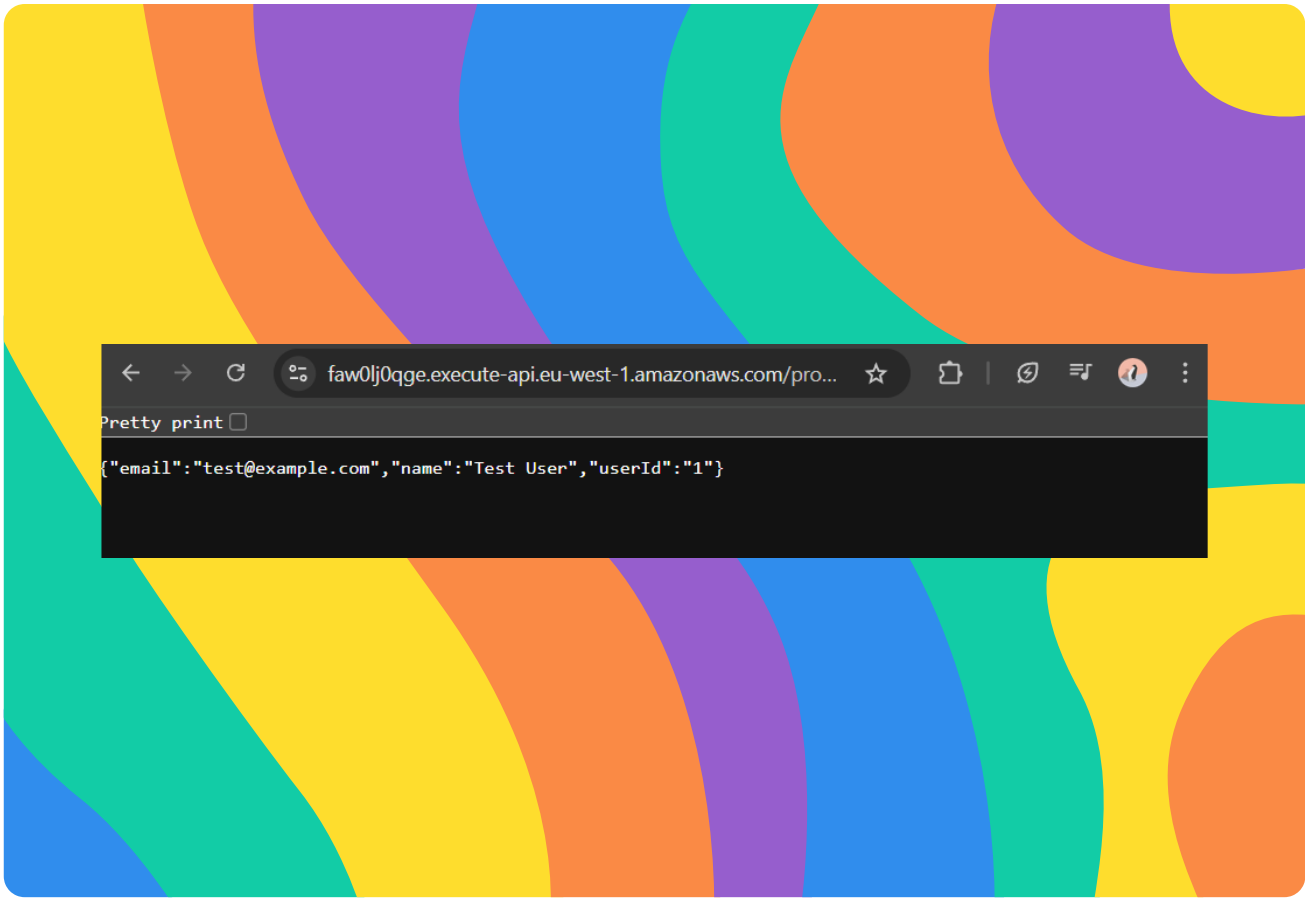


jyothesh karnam
NextWork Student

Logic and Data tier

Once all three layers of our three-tier architecture are set up, the next step is to connect the presentation and logic tier. This is because currently there is no way for our API to catch requests that users make through our distributed site.

To test my API, I visited the Invoke URL of the prod stage API. This let me test whether I could use the API and retrieve user data. The results were some user data in JSON when I looked up userId=1. This proved a logic + data tier connection.



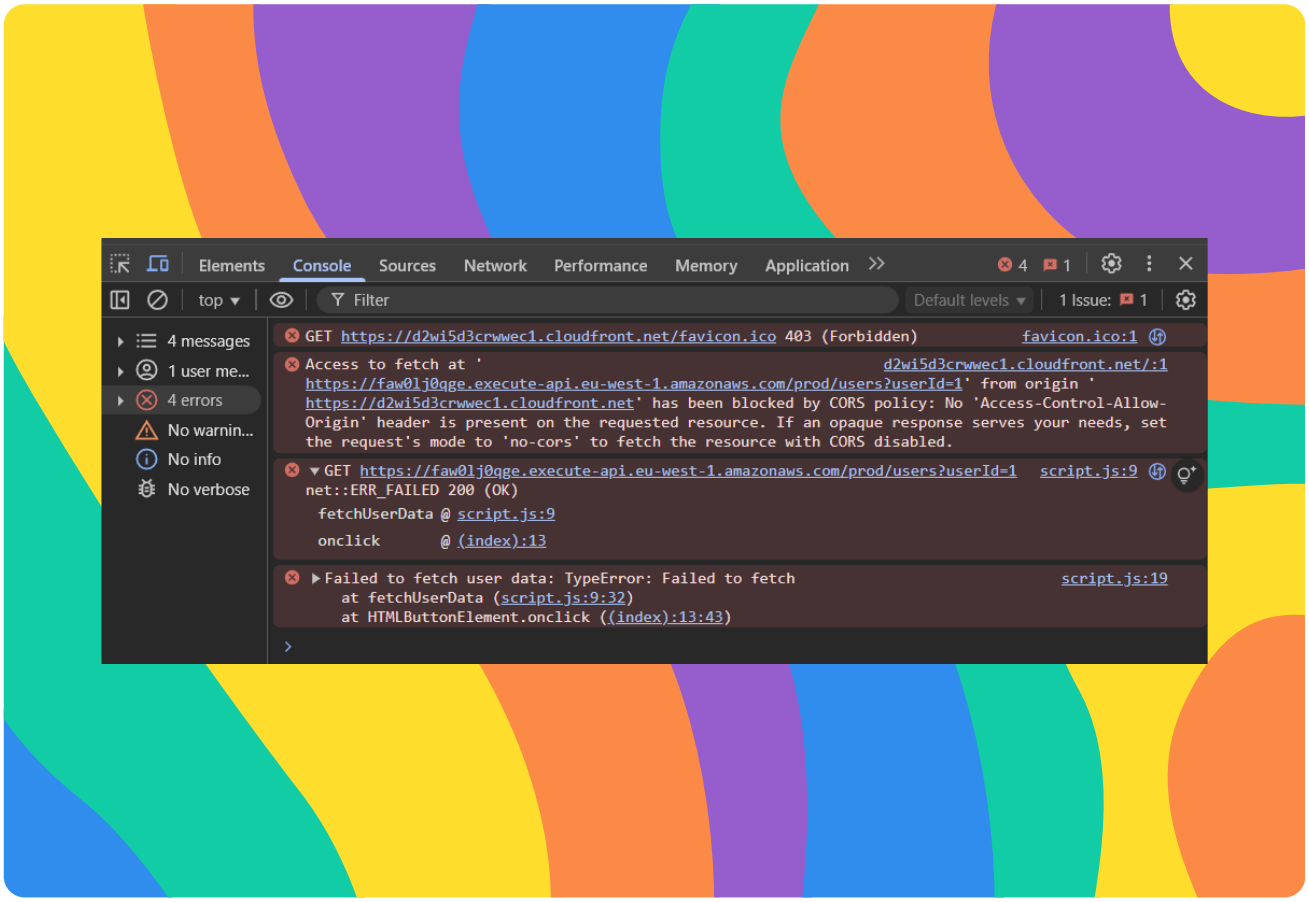
Console Errors

The error in my distributed site was due to an issue within script.js (one of the website files I had uploaded into S3). The script.js file was referencing a prod stage API URL placeholder instead of my API's actual URL.

To resolve the error, I updated script.js by replacing some placeholder text with the API's prod stage Invoke URL. I then reuploaded script.js into S3 because the S3 bucket was still storing the last uploaded version (i.e. the one with the error).

I ran into a second error after updating script.js. This was a browser-side error with CORS because API Gateway, by default, is only configured to allow requests from users directly running its Invoke URL in the browser (not with CloudFront).

jyothesh karnam
NextWork Student

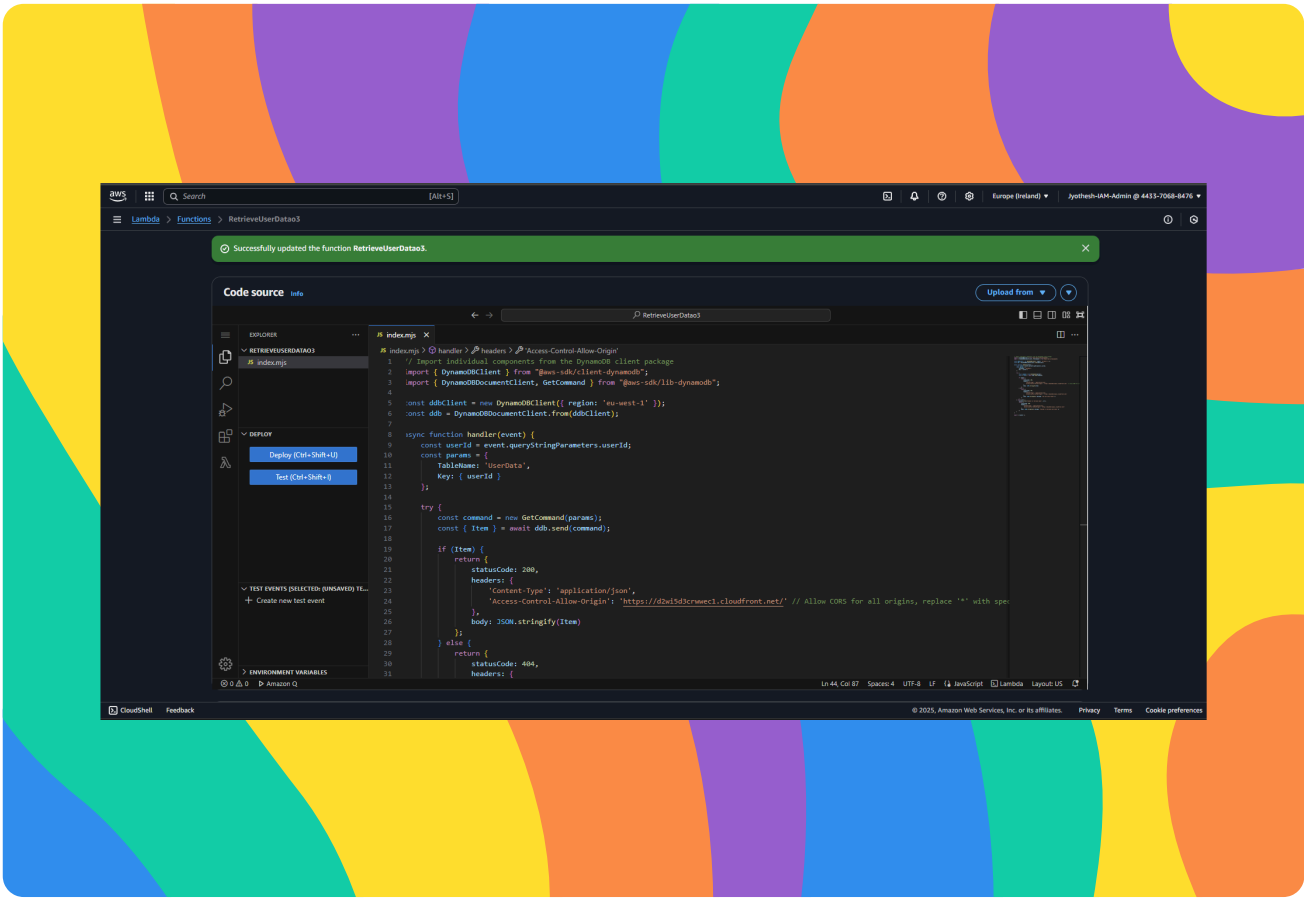


jyothesh karnam
NextWork Student

Resolving CORS Errors

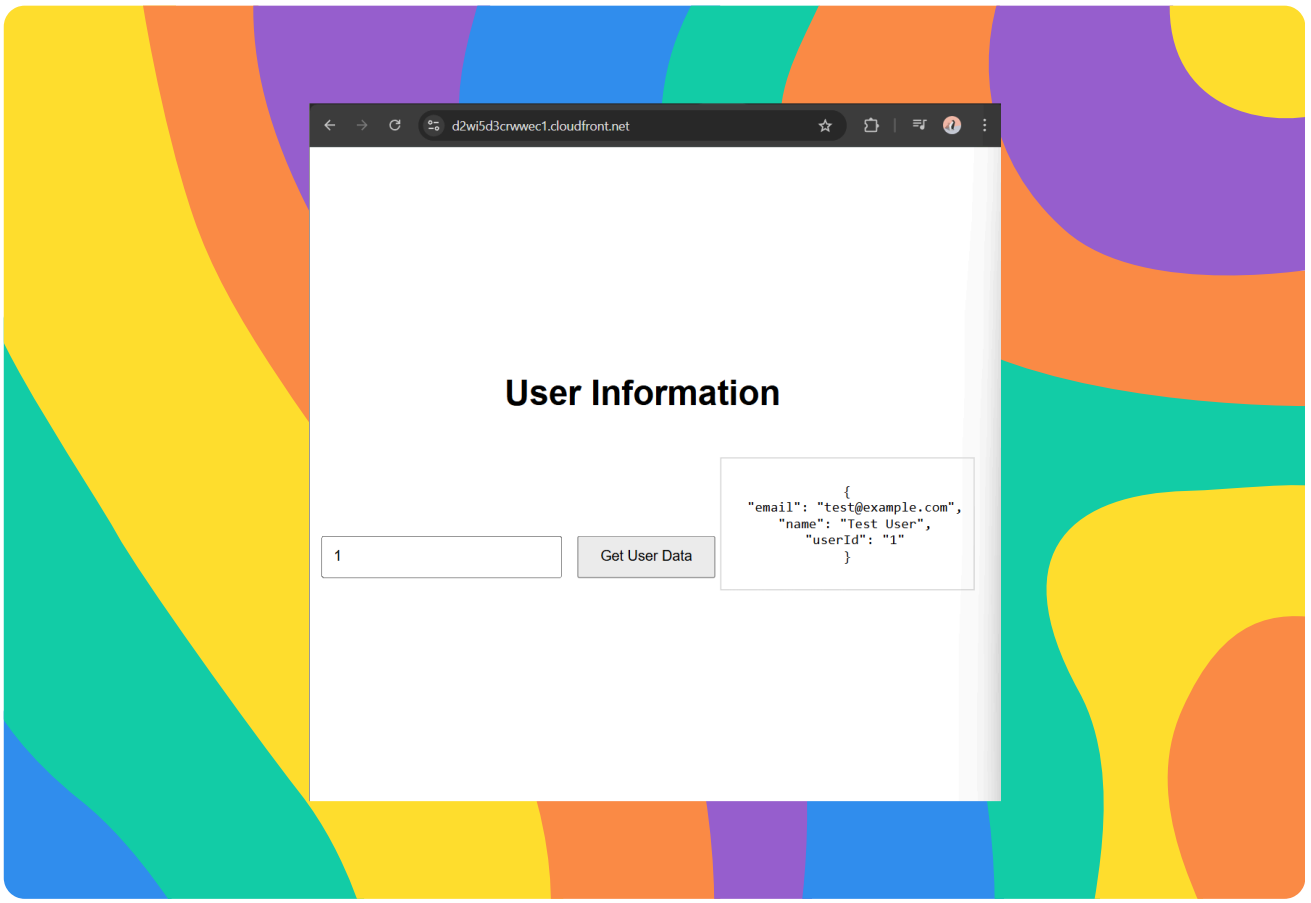
To resolve the CORS error, we first went into our API (in API Gateway) and enabled CORS on the /users resource. We then made sure GET requests are enabled, and referenced our CloudFront domain as the domain getting access.

I also updated my Lambda function code because it needs to be able to return CORS headers to show that it has the permission to invoke the API's URL and return a response. I added 'Access-Control-Allow-Origin' as a header in the response.



Fixed Solution

I verified the fixed connection between API Gateway and CloudFront by returning to the distributed site and looking up user data again. In my final test, user data could be returned – so a user request in the presentation tier gets data from the data tier.



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for more projects