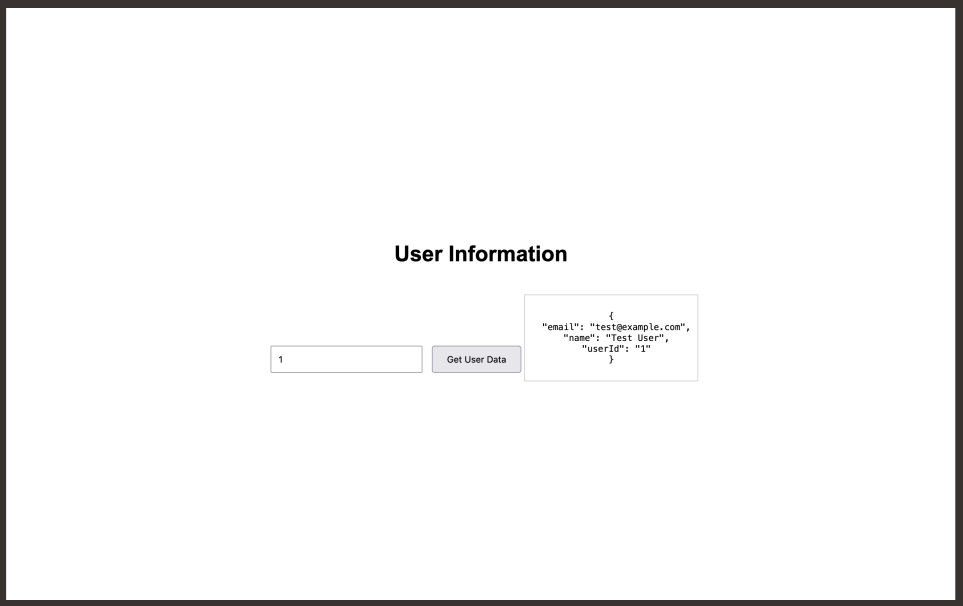
nextwork.org

Build a Three-Tier Web App

saqibh49@gmail.com




saqibh49@gmail.com
NextWork Student

Introducing Today's Project!

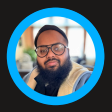
In this project, I will demonstrate how to build a fully functional web app with all 3 tiers - presentation, logic, and data. I'm doing this project to learn how web apps are deployed and managed in production environments.

Tools and concepts

Services I used were Lambda, API Gateway, DynamoDB, S3, and CloudFront. Key concepts I learnt include Lambda functions, APIs, HTTP requests, API Gateway resources and methods, API stages, Lambda proxy integration, CORS configuration, DynamoDB JSON formatting, CloudFront cache invalidation, and connecting a serverless backend to a frontend using a three-tier architecture.

Project reflection

This project took me approximately 4 hours. The most challenging part was troubleshooting the errors in the developer tools console. It was most rewarding to finally see my web app pulling data from DynamoDB.



saqibh49@gmail.com
NextWork Student

nextwork.org

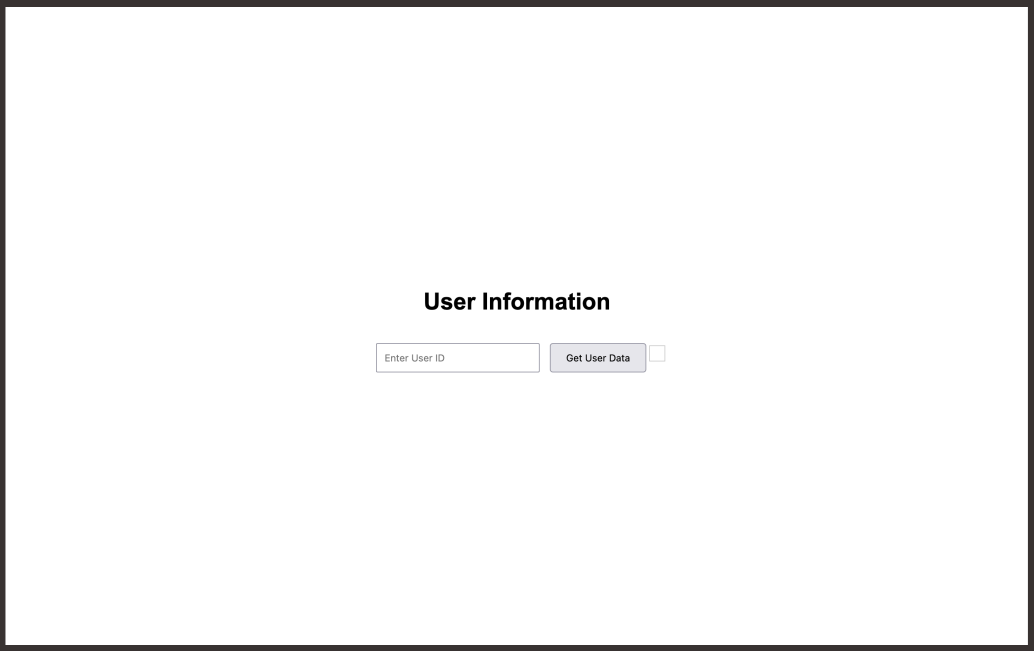
I chose to do this project today because I know the only way to get better at something is to get the reps in, and this was an enormous number of reps! Something that would make learning with NextWork even better is if I could practice the js and json code a bit more right on th nextwork site to get some practice in and get used to the syntax.

saqibh49@gmail.com
NextWork Student

Presentation tier

For the presentation tier, I will set up an S3 bucket to store my index.html file, style.css files and script.js file and set up CloudFront to make my web app available globally.

I accessed my delivered website by using the CloudFront domain that was autogenerated when I created my CloudFront distribution.

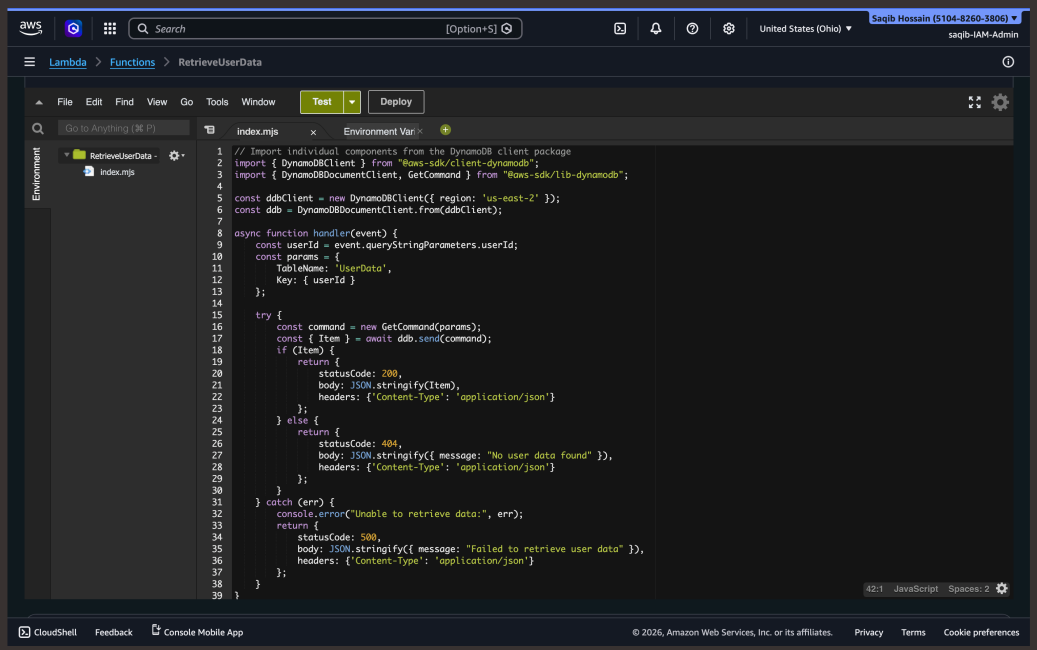


saqibh49@gmail.com
NextWork Student

Logic tier

For the logic tier, I will set up a Lambda function and API Gateway, which will connect to a DynamodB table in order to show the relevant user data on my CloudFront site.

The Lambda function retrieves data by looking for specific user data based on a userId. If it finds the userId, it returns data related to it, otherwise, it returns an error message.

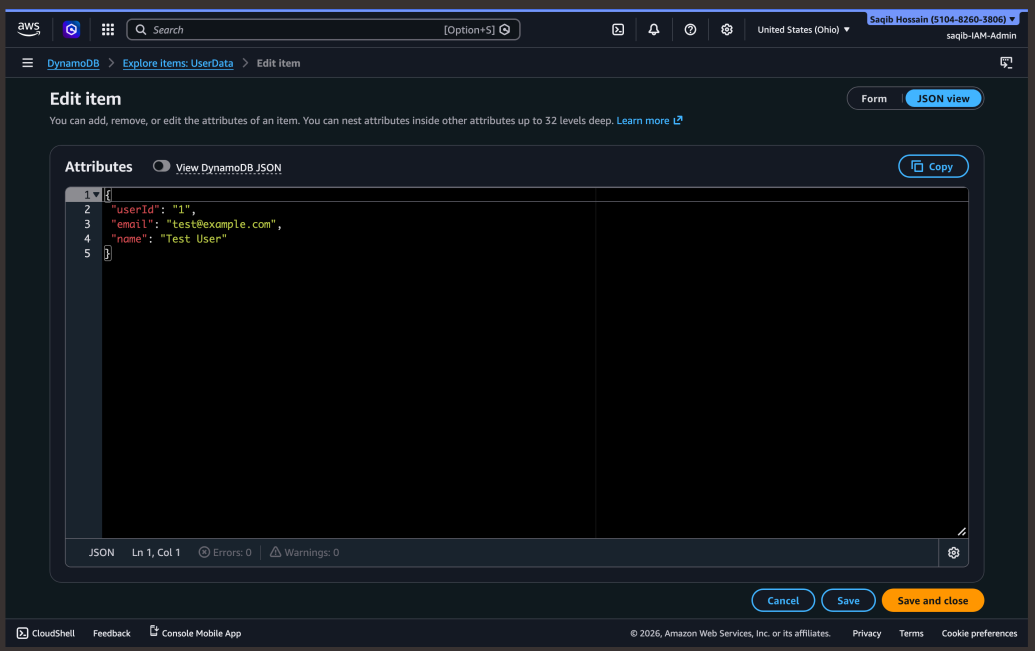


saqibh49@gmail.com
NextWork Student

Data tier

For the data tier, I will set up my DynamoDB table because this is where my Lambda function will pull user data from.

The partition key for my DynamoDB table is userId, which means DynamoDB uses that value to organize and quickly look up items in the table. Every item needs a unique userId so DynamoDB knows exactly where to find it.

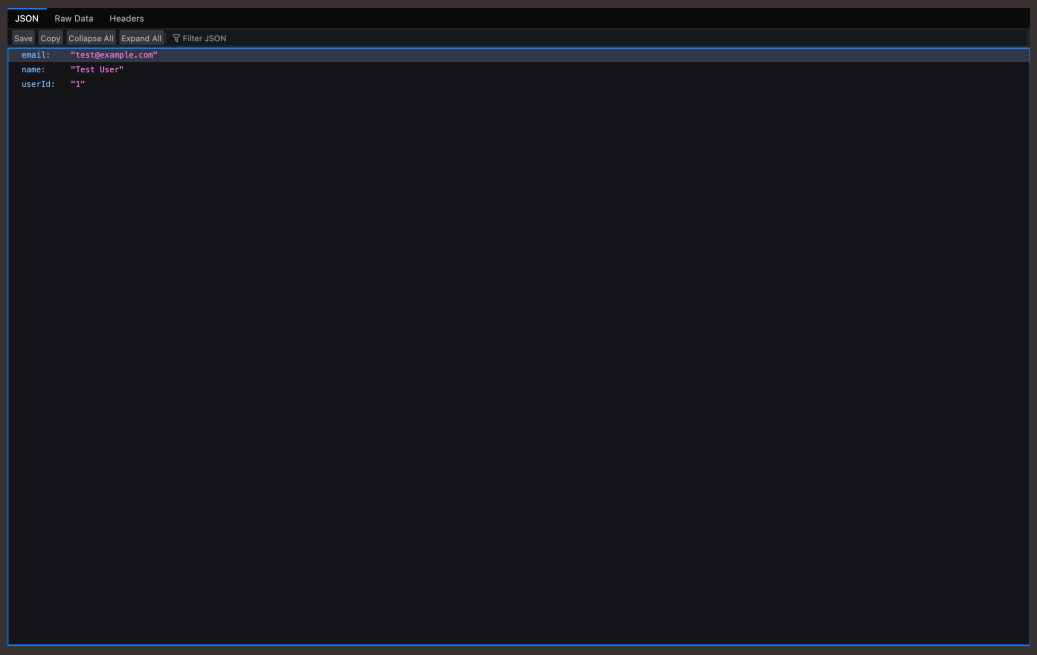


saqibh49@gmail.com
NextWork Student

Logic and Data tier

Once all three layers of my three-tier architecture are set up, the next step is to connect them by updating my script.js file to make API requests.

To test my API, I visited my invoke URL with /users?userId=1 appended to it. The results were that my API successfully returned the user data from my DynamoDB table, confirming that my Lambda function, API Gateway, and DynamoDB are all connected and working together.



saqibh49@gmail.com
NextWork Student

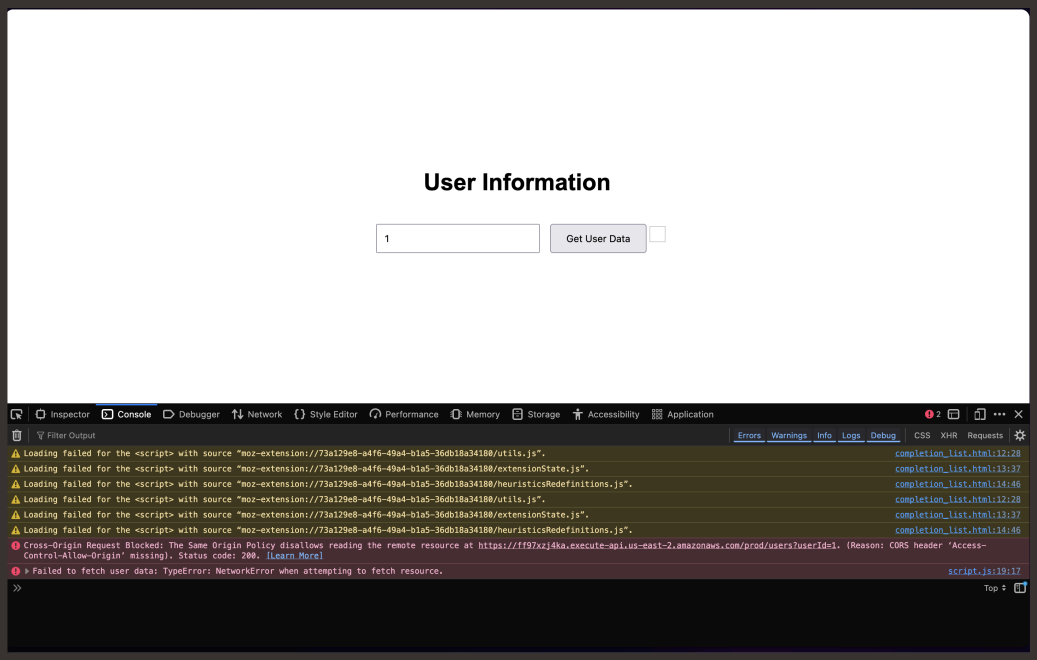
Console Errors

The error in my distributed site was because the url the button on my site is pointing to has some placeholder text where I need to add my prod api url.

To resolve the error, I updated script.js by adding my api invoke url to the button click command. I then reuploaded it into S3 because the new file contains the correct url, whereas the old file contains filler.

I ran into a second error after updating script.js. This was an error with CORS because my CloudFront URL and my API Gateway URL are two different domains, and browsers don't let websites make requests to different domains by default. Basically, my frontend and backend need to be told it's okay to talk to each other.

saqibh49@gmail.com
NextWork Student

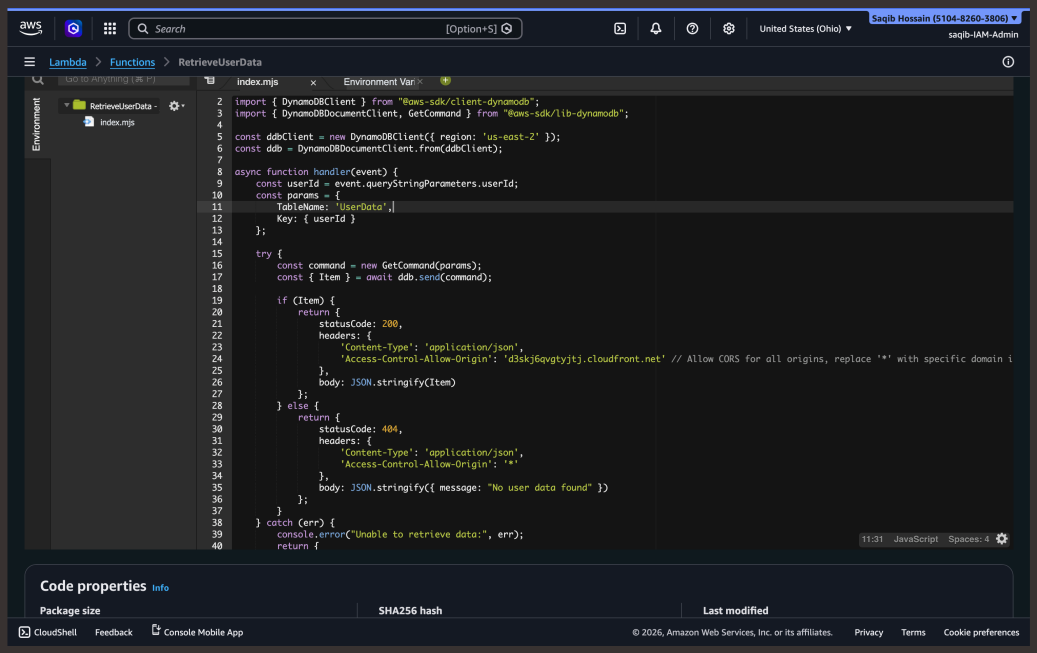


saqibh49@gmail.com
NextWork Student

Resolving CORS Errors

To resolve the CORS error, I first allowed GET permissions, then added the CloudFront distribution URL to the allow list in API Gateway.

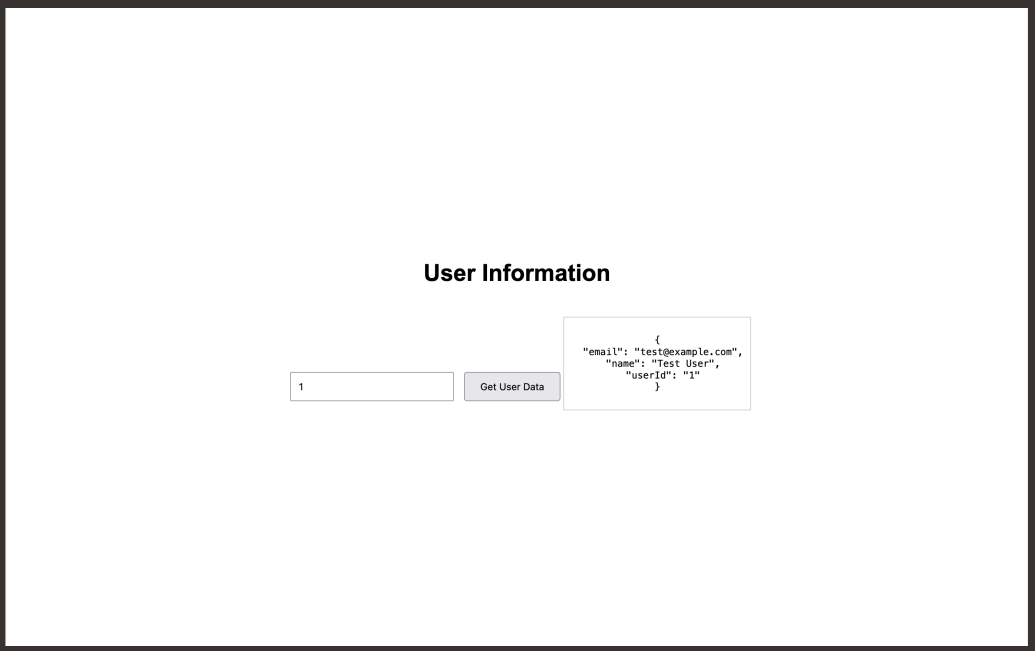
I also updated my Lambda function because without CORS headers in the response, the browser would keep blocking my frontend from receiving data from the API. The changes I made were adding Access-Control-Allow-Origin headers to all of my function's responses, which basically tells the browser that it's okay for my CloudFront site to make requests to this API.



saqibh49@gmail.com
NextWork Student

Fixed Solution

I verified the fixed connection between API Gateway and CloudFront by entering a userId in the User Information form on my CloudFront site and clicking "Get User Data." The page successfully returned the user's data from my DynamoDB table, confirming that my frontend, API Gateway, Lambda function, and database are all connected and communicating properly.



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects