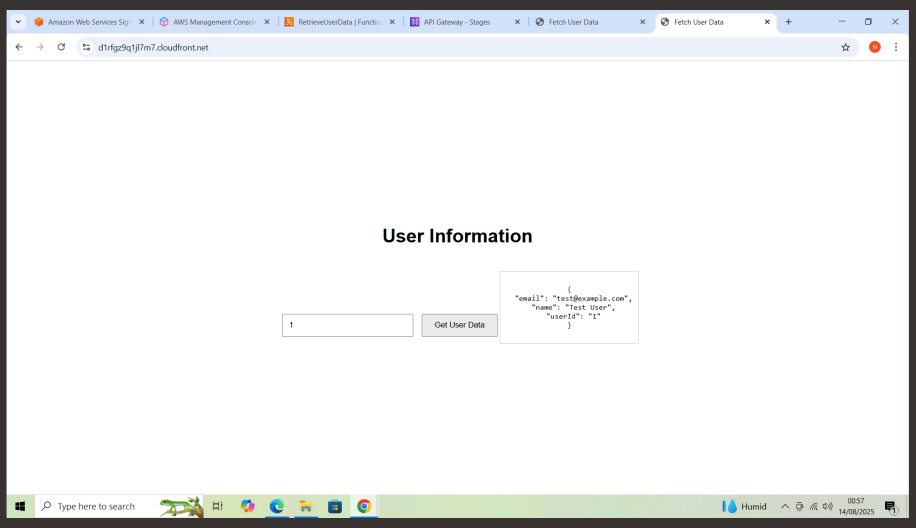
nextwork.org

Build a Three-Tier Web App

Nchindo Boris



Nchindo Boris
NextWork Student

Introducing Today's Project!

In this project, I will; - Create a storage bucket for my website's files with S3. - Distribute my content globally with CloudFront. - Build the brains of my application using serverless functions with Lambda. - Create an API to handle user requests with API Gateway. - Store and retrieve user data with DynamoDB. - Connect all these services together seamlessly for my three-tier architecture.

Tools and concepts

Services I used were; - Amazon S3 - Amazon CloudFront - AWS Lambda - Amazon API Gateway - Amazon DynamoDB Key concepts I learnt include; - Store website files with S3. - Deliver content globally with CloudFront. - Write serverless code with Lambda. - Create and manage APIs with API Gateway. - Store and retrieve data with DynamoDB. - Connect all these services together to build a fully functional three-tier web application.

Project reflection

This project took me approximately 4 hours to complete. The most challenging part was connecting the 3 tiers. It was most rewarding to see the data succesfully fetched from DynamoDB displayed on your website!

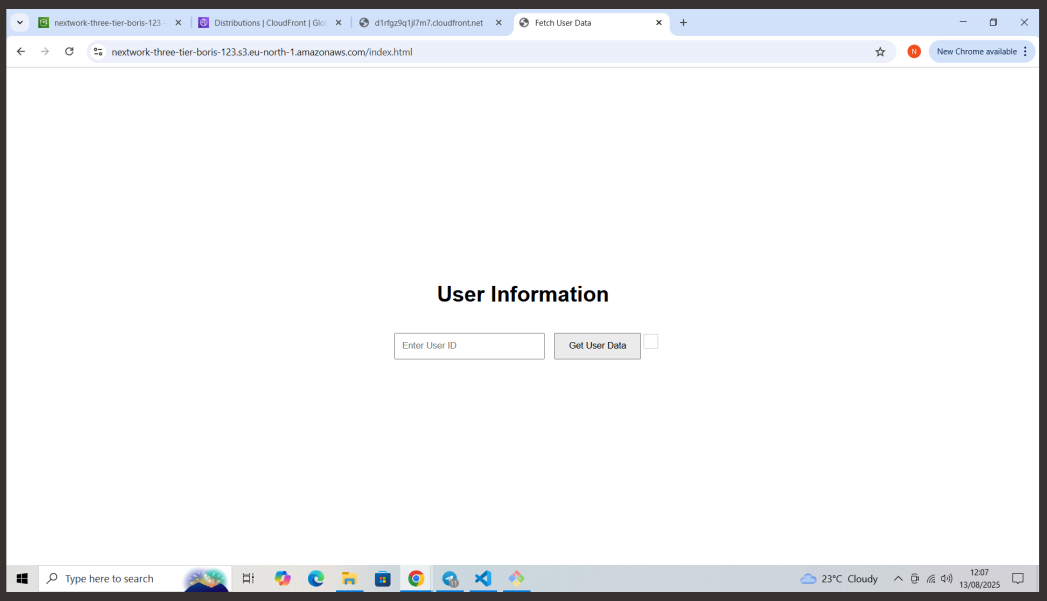
Nchindo Boris
NextWork Student

I did this project today as it is one of the essential projects and goals in getting a cloud career

Presentation tier

For the presentation tier, I will; - Create an S3 bucket to store my website's files. - Upload a simple index.html file to my bucket. - Set up CloudFront to deliver my website's content globally.

I accessed my delivered website by; - Heading back into my CloudFront console. - Copy the distribution domain name - Paste the domain name into your web browser.

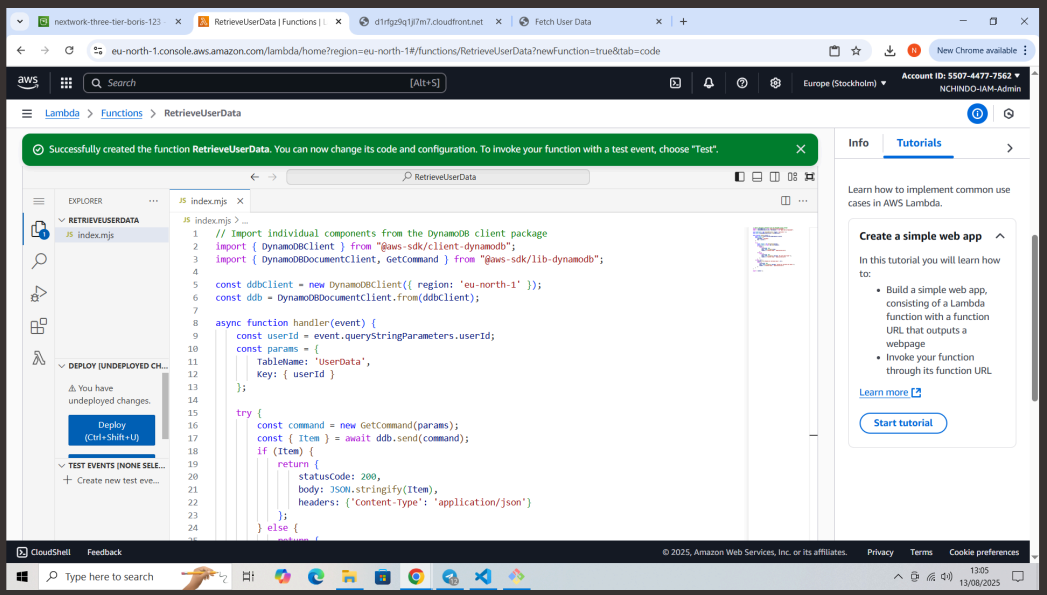


Nchindo Boris
NextWork Student

Logic tier

For the logic tier, I will set up; - Create a Lambda function to fetch data from a DynamoDB table. - Write the code for my Lambda function. - Create an API Gateway REST API. - Create a resource and method to handle GET requests. - Deploy the API to make it accessible.

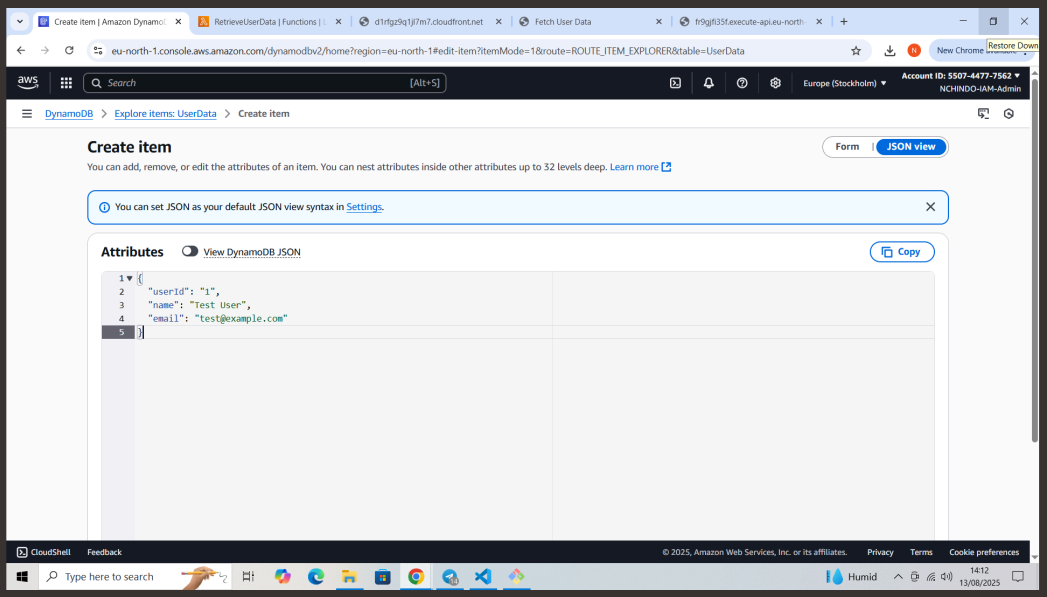
The Lambda function retrieves data by looking for specific user data based on a userId and returns that data.



Data tier

For the data tier, I will set up; - A DynamoDB table. - Add user data into my table.

The partition key is userId which means all items for users should be stored in the same partition

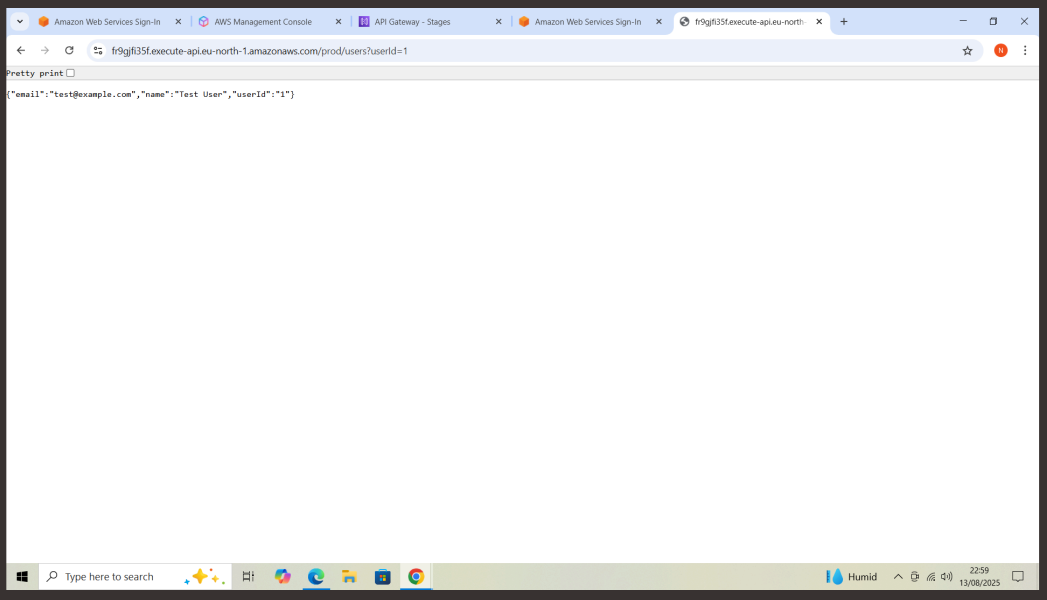


Nchindo Boris
NextWork Student

Logic and Data tier

Once all three layers of my three-tier architecture are set up, the next step is to; - Update my script.js file with JavaScript code to make an API request. - Verify that the data is displayed on my website.

To test my API, I Copied my prod stage API's Invoke URL, appended /users?userId=1 to the end of the URL I copied and then Ran the edited URL in my web browser. The results were my table's data getting returned by the API.



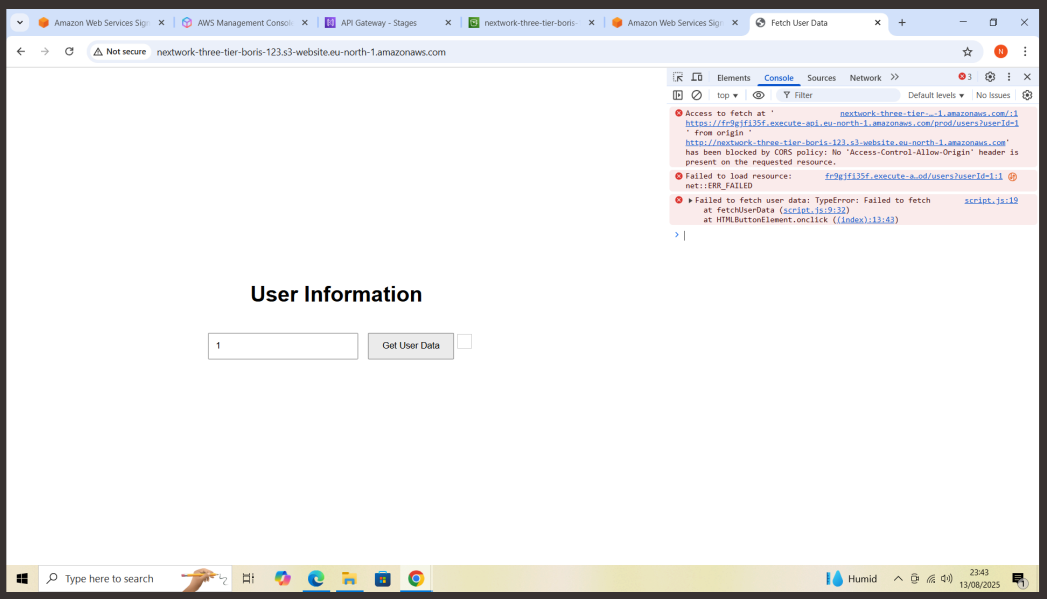
Console Errors

The error in my distributed site was because of lack of connection between the presentation and logic tiers which is what connects my website to my Lambda function

To resolve the error, I updated script.js by; - Opening script.js in Notepad and I see there's a line that directly references [MY-PROD-API-URL] - Find and Copy your prod stage API's Invoke URL in the API Gateway console. - Paste this in script.js, making sure to replace [MY-PROD-API-URL] in the script. - Save your changes. I then reuploaded it into S3 because it is needed to create a connection between the presentation and logic tiers

I ran into a second error after updating script.js. This was an error with CORS (Cross-Origin Resource Sharing) because my API Gateway is not configured to allow requests from my CloudFront URL. API Gateway is only allowing requests directly from its Invoke URL! To resolve this, I need to enable CORS on your API Gateway so that it can accept requests from the domain where my frontend is hosted.

Nchindo Boris
NextWork Student



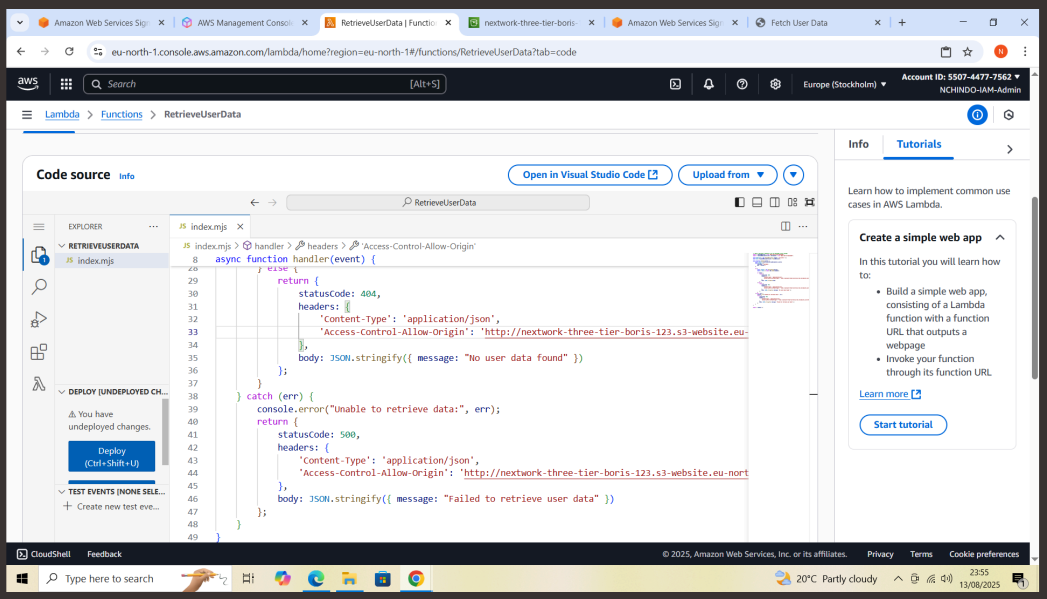
Nchindo Boris
NextWork Student

Resolving CORS Errors

To resolve the CORS error, I did; - To Amazon API Gateway console in your AWS account. - Navigate to the Resources tab. - Select the /users resource. - Select Enable CORS - In the CORS configuration, check both GET and OPTIONS under Access-Control-Allow-Methods. - Enter your CloudFront distribution domain name as the Access-Control-Allow-Origin value. This will allow requests from your CloudFront domain to your API. - Select Save.

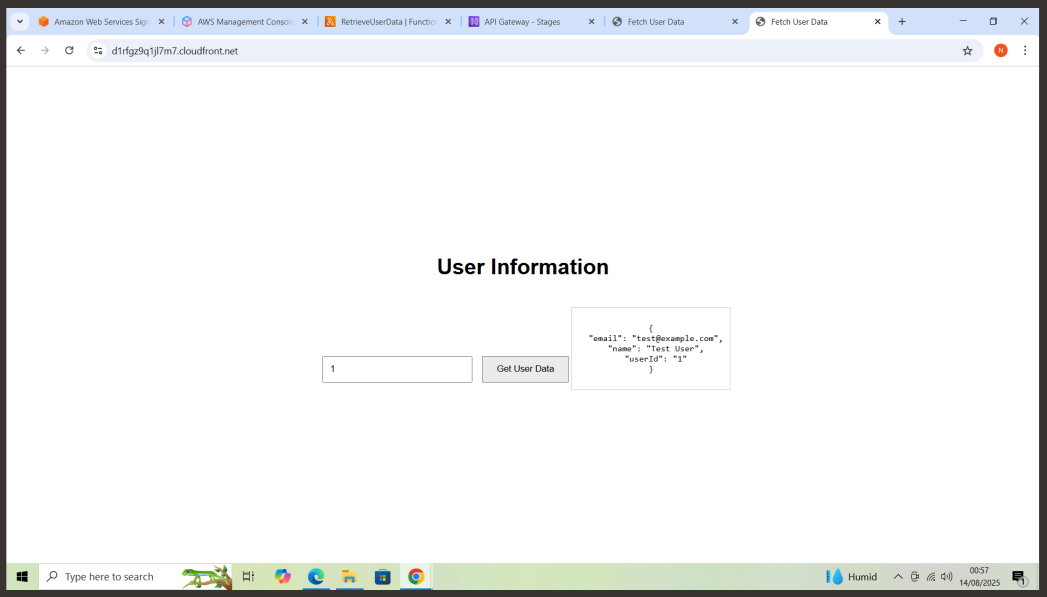
I also updated my Lambda function because After enabling CORS, you must redeploy your API for the changes to take effect. The changes I made were; Updating my Lambda function code to include the Access-Control-Allow-Origin header in the response:

Nchindo Boris
NextWork Student



Fixed Solution

I verified the fixed connection between API Gateway and CloudFront by refreshing the CloudFront domain name now saw the data fetched from DynamoDB displayed on your website!



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects