Fraud Fighters: Using Serverless Architecture and Next.js to Fight Fraud

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Jeffrey Bukont

Spring, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Fraud Fighters: Using Serverless Architecture and Next.js to Fight Fraud

CS4991 Capstone Report, 2022

Jeffrey Bukont
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jabukont@gmail.com

ABSTRACT

Capital One manages fraud by creating concerns on credit cards it suspects might be used for fraud. In their previous system, concerns needed to be created manually, which required filling out information using a form. As summer interns, we were tasked with creating a system that would automatically import account information, speeding up concern creation. We decided to leverage AWS Lambda, S3, and Next.js to build and deploy a web application which integrated multiple APIs to gather account and credit card information. Our project decreased the time to create a concern by over 90%. Although we successfully developed and deployed our project, in the future we would like to make the onboarding process for new teams more fluid.

1. INTRODUCTION

Credit card fraud is a multibillion-dollar criminal industry that affects millions of Americans each year. The costs of most fraudulent transactions are not passed onto the credit card holder, but instead onto the bank which issued the credit card. The cost of these transactions makes limiting fraud a top priority for credit card companies, because reducing the number of fraudulent transactions by a few percentage points can result in tens of millions of dollars saved each year.

This summer I worked at Capital One on a project which reduced the time of the manual concern creation process by 90%. This increased speed should decrease the time required to test new fraud fighting innovations and save the company money.

2. RELATED WORKS

Maniraj, et al (2019) recommend using machine learning to flag fraudulent transactions. This is how most transactions are flagged. However, he notes that in specific scenarios like testing environments, a user might need to create a concern on a credit card manually. This is similar to Capital One's environment were for commercial customers concerns are created using A.I, but on a testing and development side concerns are created manually.

In their case study, Choudhary, et al (2020) discuss building a serverless chat application using a combination of AWS Lambda, s3, and DynamoDB. Their process was very similar to ours except that we did not use DynamoDB.

3. PROJECT DESIGN

For our web application my team chose to work in Next.js because our manager tasked us with using server-side rendering, server-side rendering moves some of the burden of webpage rendering from a user's computer to a server. Server-side rendering helps to standardize performance across a wide range

of devices, which is optimal in a business environment. NextJS comes with this feature built in through a function called Server-Side Props, so we thought it was an obvious choice.

The frontend was essentially ReactJS since NextJS is built upon React. We used MaterialUI which is a react component library. From this we developed a UI where the user could search or select a list of credit cards associate with their TDM account. Then after clicking on a credit card an API call was made to fetch all transactions which were then displayed on the UI. After clicking on a transaction, the User could create or delete a concern associated with that transaction. Then the user would receive a notification that the API call to delete or create a concern was successful or failed.

We used a combination of AWS Lambda and S3 to avoid using EC2 servers because a serverless architecture (Lambda + S3) would increase uptime and decrease maintenance costs for our application. AWS lambda is a tool that allows code to be run on a server in response to a certain event happening. For example, if a user clicks the upload button on a webpage, then Lambda can run the code to upload the data to the database. However crucially, Lambda is not a server since the owner does not have to worry about provisioning compute power or memory. Lambda runs code on servers managed by AWS, and are billed for how long and often code runs. This is great for web applications which will be used with varying frequency but require high levels of uptime to be useful.

S3 is a storage service that allows user to store files in buckets. We used S3 to store the frontend of our website in a bucket and made the bucket accessible by URL using another AWS service called route53. This meant when the user went to the URL the HTML files were rendered by the user browser. When the user clicked on a button that required fetching new information, Lambda would be notified and send back the relevant data.

Last, we integrated the Capital One's account and credit card information fetching APIs using NodeJS. We finished our application but ran into problems with server-side rendering during deployment. We needed to use the Serverless Next.Js framework to deploy to AWS Lambda, but Capital One's pipeline had not approved the serverless framework, so we had to change to Client-Side Rendering.

4. Results

After switching to client-side rendering our project worked very well, decreasing the time to create a concern by over 90%. Our manager even said our intern project would be built upon by a full-time team after we left.

5. Conclusion

Overall, our app increased the efficiency of testing Fraud Fighting applications by making it easier to create concern on test account. In the long run this increased efficiency will lead to more tools to help combat credit card fraud, which is a multibillion-dollar criminal industry.

6. Future Work

One aspect of our app that could have been improved was the onboarding process for new Capital One teams. Almost all our API calls were to Capital One's intern Team Database Management service, which meant we needed some way of adding new team ClientID's and Client Secrets. We came up with a solution, but it required new team to manually add their ClientID's and Client Secrets to Capital One's credential database called Chamber of Secrets. This process is somewhat tedious so an improved version of our web application should have added a built-in signup and login page for Capital One teams.

7. UVA Evaluation

The class that most prepared me for this internship was Advanced Software Development Methods. The class and my internship were extremely similar as both were group projects based around developing a web application from the ground up.

One aspect UVA could improve upon is more assignments where you must analyze or improve someone else's code. This is a valuable skill that came up multiple times in my internship, but I felt I didn't experience in many of my classes. Additionally, classes that incorporate the deployment process would be very beneficial. However, I understand that this process would be harder to incorporate into curriculum. Which is why interns from various schools struggled with the deployment aspect of the internship.

REFERENCES

Maniraj, S. P., Aditya, S., Shadab, A. & Swarna, S. (2019). Credit Card Fraud Detection using Machine Learning and Data Science. International Journal of Engineering Research and. 08. 10.17577/IJERTV8IS090031.

Choudhary, B., Chinmay, P., Aditya, G., Ankit, D. & Shilpa, S. (2020). Case Study: Use of AWS Lambda for Building a Serverless Chat Application. 10.1007/978-981-15-0790-8_24.