Technical Report Title

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

Quentin Bishop

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

Rosanne Vrugtman, Department of Computer Science

Designing an Admin Dashboard for an Anti-Money Laundering Platform

CS4991 Capstone Project, 2021

Quentin Bishop
Computer Science
University of Virginia
School of Engineering and Applied Science
Charlottesville VA USA
qjb8jg@virgina.edu

ABSTRACT

Money laundering is a financial crime often used to fund criminal or terrorist organizations that often passes through banking systems completely unnoticed. Last summer I participated in a software engineering internship at a bank in which I worked with a team developing an anti-money laundering application to help detect and report suspected cases of money laundering. More specifically, I worked with a team to create an admin module to help manage money laundering investigations in the bank's internal money laundering detection system. During this internship, I worked heavily with the Go programming language, Git for collaboration and code management, AWS serverless functions, Agile development methodology, and many DevOps tools. By the end of the internship, my team had successfully developed an admin dashboard that could fetch data about ongoing money laundering investigations and display important information for administrators of the anti-money laundering platform. In the future, more data needs to be fetched and data visualizations need to be incorporated to more effectively present the retrieved data.

1 INTRODUCTION

In recent years, an explosion of cyber crimes has taken the public by storm. From countless classified government documents being hacked and posted on Wikileaks to millions of stolen credit card numbers being listed for sale on the dark web, it seems cyber crimes have become omnipresent in our society. Cyber crimes with a financial component are especially troubling as stolen money can be used to fund drug cartels, terrorist organizations, or even rogue states like North Korea. However, money obtained from any sort of criminal activity is often laundered through banks to hide its illegal source.

With millions of transactions taking place every day online, an enormous amount of data is generated that is nearly impossible for humans to analyze. As a result, many cases of money laundering go unnoticed simply by hiding in plain sight within the enormous amount of data. To counter this, banks have started to develop automatic money laundering detection systems that harness the power of artificial intelligence to sift through millions of transactions and flag suspected cases of money laundering. However, many of these programs are still in their infancy and not yet fully automated. Current algorithms may flag benign transactions while failing to flag obvious cases of money laundering.

Since current algorithms are not yet sufficiently advanced to accurately automate the entire process, a hybrid technique is used in which algorithms flag many suspected cases of money laundering, and a human investigator manually analyzes each suspected transaction to determine whether the transaction is in fact money laundering. This hybrid solution alleviates the hassle of investigators needing to analyze millions of benign transactions looking for money laundering while also not entirely handing control to an algorithm that is not fully reliable.

In order to manage a complex platform that relies both on algorithms and investigators, my team created an administrator dashboard that shows analytics and relevant information about investigators and investigations such as how many investigations have been completed in a certain time frame, how many investigations an investigator has completed in a certain time frame, how many transactions have been flagged and other data that helps in managing the platform. Previously, analytics for the platform needed to be generated manually and were only done ~~so~~ once a year because of the extensive work required. The admin dashboard we built alleviates the issue of manually generating analytics and reports regarding investigations by continuously and automatically generating live analytics on a dashboard that platform administrators can view and access at any time.

2 RELATED WORKS

Admin dashboards are prevalent on many sites, but perhaps the most effective and well-known is the YouTube creator dashboard. YouTube uses a dashboard that effectively displays all relevant analytics for an admin user while also incorporating aesthetic and interactive data visualizations. YouTube fetches relevant data from all of a user's videos and aggregates it to calculate and display them to the user. Interactive data visualizations mean that a user can hover their cursor to view more detailed information about the data. Additionally, the YouTube creator dashboard contains various tabs, each containing different categories of data such as views, watch time, audience engagement, audience retention, and advertisement revenue [1]. In a similar vein, our anti-money laundering platform's admin dashboard aggregates data from all relevant investigations and

displays it to the user with interactive data visualizations. Additionally, we created various tabs on our admin dashboard, each containing various categories of data.

3 PROJECT DESIGN

The admin dashboard consists of an admin module user interface that calls AWS lambda serverless functions. The dashboard also uses functions from a custom library called Golib and is deployed using various Devops tools.

3.1 Admin module UI

The first component in developing our admin dashboard was creating an admin module within the existing User Interface (UI). The existing anti-money laundering platform uses an Angular framework which is based Typescript. I created a new module to house the admin dashboard and created a temporary placeholder page that contained the text "Future home of the admin dashboard." I then modified the routing module within angular so that adding "/admin" to the URL would route a user to the placeholder page I had created. I then submitted a request to the owner of the anti-money laundering platform to create an admin role which had exclusive permissions to visit the admin dashboard page, meaning another user would be denied access. After my request was approved, I modified the routing module once again so that only users with the admin role would be able to route to the admin landing page.

My team then worked to develop Typescript functions that would call queries and return the appropriate data to be displayed on the page. These functions would take in any user input needed for the query, such as a date rage, and call the appropriate AWS function which returns the data to be visualized. Because of the scale of the anti-money laundering platform, we often had to submit design reviews to the design team and get approval before we could publish our changes to the UI. This ensured that the style throughout the anti-money laundering platform was consistent and intuitive. Occasionally, our design reviews would be rejected or the design team would ask us to make modifications before proceeding with additional changes. We also began working on data visualization, but unfortunately, we had not finished by the end of our internship so we were unable to deploy data visualizations to the admin dashboard.

3.2 Golib

An important step in developing our admin dashboard was creating our own library of functions that could be used throughout development. As most of our project used the Go programming language, we decided it would be efficient to store all shared functionality in one central library instead of rewriting or copying and pasting code wherever it is needed.

The first module we created within our shared library was a simple logger package. Logging is essential in development as it allows programmers to determine what components worked and what went wrong. Our logging package logged all errors and outputs as well as where they took place so that debugging errors during testing would be easier. We also created an environment package which allowed us to deploy our admin dashboard to various environments for various of development. Having multiple environments to deploy to helps alleviate the risk of a small change crashing the entire system. With multiple environments, we can test all of the changed we made in a quarantined environment and see how our changed affect the rest of the anti-money laundering platform before we publish our changes to the production version of the platform currently in use. Finally, we created a database package that simplified the way we connect and communicate with the databases. Instead of having to go through the tedious steps of setting up a secure database connection any time we want to query data, we can simply call the functions in our database package to set up the connection for us.

3.3 Serverless Functions

Another key component in developing our admin dashboard was the serverless functions. We used Amazon Web Services (AWS) lambda functions written in Go to perform all the queries for our admin dashboard. Lambda serverless functions are a type of AWS service that allows user to make queries without having a server continuously running. As a result, lambda functions are much more cost effective than a typical AWS server such as an EC2. Lambda functions can be written in a variety of languages, but we decided to write our queries in Go as that was the language being used throughout the anti-money laundering platform. Our serverless function queries received input from the UI and then sent a request through AWS to access our investigations database. We'd then use packages from the golib we created to interact with the data and retrieve the relevant information for the query.

At this point, our lambda function could perform the appropriate calculations to generate statistics and return them to the user. Because the serverless functions involved many different systems working together seamlessly, we performed extensive testing at various stages of development starting from returning a simple "Hello World" response to a query all the way up to retrieving database information and calculating the dashboard statistics all in one function.

3.4 Devops and Integration

Finally, we had to use various devops tools including Jenkins and Bogie to build and deploy our application. Since the entire anti-money laundering platform deals with sensitive financial information, it is paramount that all code is secure and thoroughly tested. To ensure that all code is thoroughly tested, we designed test cases for every function to ensure that we had 100% code coverage. A strong code coverage ensures that all code works as intended and helps to minimize and vulnerabilities. Once we had developed test cases for all of our code, we needed to test it at every stage of development.

In order to ensure continuous testing and continuous integration, we created a pipeline in Jenkins that would run all of our test cases every time we updated the code to ensure than new changes didn't impact preexisting functionality. In addition to ensuring all the tests passed, our Jenkins pipeline also checked for poorly written code and linting errors, and deployed to various stages of development. We built our pipeline so that we could deploy to any environment from our golib package and manually interact with our updated dashboard to look for

bugs or errors.

4 RESULTS

In its current state, the admin dashboard we developed allows administrators of the anti-money laundering platform to view analytics regarding investigations on the platform. The admin dashboard is being used by managers to assess the productivity of investigators and help assess the accuracy of current money-laundering detection algorithms. As a result, a manual end-of-year report no longer needs to be generated. All of the data is readily and continuously accessible through the admin dashboard. The automated analytics also allow for the more rapid development of money laundering detection algorithms since their success can be evaluated continuously instead of having to wait for the end of year report to determine how many flagged transactions were indeed cases of money laundering. Since the admin dashboard helps expedite the process of detecting and investigating cases of money laundering, it results in more cases of money laundering being caught and prosecuted and thus helps reduce the number of financial crimes being committed.

5 CONCLUSION

Detecting and punishing financial crimes continues to be a difficult task in a digital age where fraudulent or criminal transactions can go unnoticed simply by hiding in plain sight within the extraordinary amount of data being generated. While the battle against money laundering is ongoing, the admin dashboard we developed will help in expediting the detection and reporting of suspected cases of money laundering. The admin dashboard was also built as a framework with the potential to easily add more analytics and tools for administrators. Additionally, working on such a cutting edge project exposed me to many tools and industry “norms” that are essential to learn as a software engineer.

Having studied computer science at UVA, I found Software Development Essentials (SDE) to be the most useful course in preparing me for my internship experience. It taught many essential skills that are commonplace throughout the tech industry such as Git, database management, and Agile development. While SDE was certainly useful, the issue arises from the fact that it is the only CS class I’ve taken at UVA that teaches material that is essential and currently being used throughout the software engineering industry.

Additionally, the UVA CS curriculum doesn’t place much emphasis on security or testing. Students can opt to take a cybersecurity class on its own, such as in Intro To Cybersecurity, but those classes are not graduation requirements and many students never take them. Additionally, the cybersecurity classes at UVA tend to place more emphasis on offensive cybersecurity techniques and exploiting vulnerabilities whereas industry relies on software engineers having a strong grip on writing secure code and mitigating risks.

In my opinion, the UVA CS curriculum could be improved by teaching newer industry-applicable skills such as Git, DevOps, and AWS early on in a student’s course of study as well as integrating cybersecurity as a theme throughout all CS classes so that students have a strong foundational knowledge of industry-prevalent tools and practices when heading into internships or jobs

6 FUTURE WORK

While the admin dashboard currently displays some analytics regarding investigations, it’s designed in such a way to facilitate adding additional queries to fetch more analytics with ease. Additionally, analytics are currently displayed on the screen in plain text instead of using interactive data visualizations. While we explored and experimented with several data visualization libraries in Angular, we ultimately were not able to integrate them into our admin dashboard by the end of our internship. Integrating a visualization of our analytics into our UI would be relatively simple since the library we were working with, ngx-charts, offers a plethora of premade customizable charts and graphs.

REFERENCES

[1] Google. 2021. Navigate YouTube Studio. Retrieved from https://support.google.com/youtube/answer/7548152?hl=en&ref_topic=9257787