# Streamlining Help Request Workflow with Slack Automation Tools

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

**Sam Gallettta**

Spring, 2023

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

# Streamlining Help Request Workflow with Slack Automation Tools

CS4991 Capstone Report, 2023
Sam Galletta
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
sjg7egt@virginia.edu

**ABSTRACT**

A Northern Virginia based event-management software company needed to replace its inefficient and unorganized process for handling internal login help requests. To streamline the process and make it more intuitive, our development team implemented some of Slack's built-in automation tools and integrated a Slack Bot supported by AWS services. Slack offers a suite of automation tools through Workflow Builder which allowed us to add forms and automated messages to our help request workflow. We also used a Slack Bot to streamline the workflow by adding a functionality enabling users to retrieve general information about our service through slash commands. The bot was supported by AWS resources including Lambda, API Gateway, and DynamoDB and the whole stack was deployed using CloudFormation. There was almost instant impact from the addition of these tools to our help request Slack channel as the developers on my team were able to handle requests at a faster rate and in a more organized manner. In addition, people asking for help about another service were deflected to the proper teams through the automated messages. In the future, we want to expand the functionality of the Slack Bot, creating a larger suite of slash commands. We also want more integration with our other tools like our Jira ticketing system and API logs.

## 1. INTRODUCTION

Fixing problems in a timely manner is an essential part of running any business, but especially in the fast-paced event management industry. Our company had no standard help request workflow, as each team had their own Slack channel to field help requests. Our team owned the company's login service, which includes both internal logins from employees and logins from our customers using our products. With such a high volume of users, there is nearly a constant stream of problems that they may run into, and we fielded those help requests through our Slack channel.

Our help request channel was a free-for-all when people needed to submit a request. Users could only submit requests through free form messages, which often led to unnecessarily wordy help requests, making problems even more confusing. The lack of forms in our workflow left the amount of information in the help request up to the user's discretion. This often led to users omitting necessary information, which made a simple problem take dozens of messages to figure out. Additionally, users often came into our channel asking for help about another service. Deflecting them in the right direction was another task added onto the plate of our already very busy developers. We knew that we needed some sort of automation integrated into our help request workflow, as there was always the need for human intervention in our process.

To put it simply, our help request system was wasting the time of both our

engineers and users. Developers would get stuck in long message threads to solve trivial issues with our service. Our users would get frustrated when they learned how simple the answer to their question was and how long it took them to get there. We knew that changes had to be made which is why we undertook this project of integrating Slack automation tools.

## 2. RELATED WORKS

Slack blog posts were some of the first resources we looked at when trying to find inspiration for our project. We landed on a post by Haughey (2019) which was almost exactly what we were looking for. In his post, he introduces a problem nearly identical to ours along with how Slack automation tools can be a solution. According to a study cited by Haughey, after surveying hundreds of help desks, they found that when given the option, 70% of employees would rather submit their help requests through Slack. This gave us peace of mind that our users would still be satisfied with keeping our help request system on Slack, we just needed to improve the process. One of the Slack tools that Haughey introduces in his post is the slash command. The slash command is message that starts with a / followed by the actual command. When sent in a Slack channel, it will perform some action. We this was something we would want to adopt in our new workflow due to its simplicity and wide range of uses.

A Medium blog post by Pilvelis (2020) was another inspiration behind our project. This post was a tutorial on how to stand up a Slack Bot backed by AWS resources. His post helped us realize the simplicity of the set-up process. Almost all our company's microservices were backed by AWS so our development team was already very familiar with similar processes. Pilvelis offered a very simplified version of what we were going to need to do as he was developing his bot for personal use, and we needed to develop one for corporate use. This meant that we would need to take further steps in ensuring the security of our bot as well as abiding by company development standards. We would ultimately have to end up making quite a few tweaks to the process that Pilvelis lays out, but his post was a necessary kick starter for our project.

## 3. PROCESS DESIGN

This section will detail the design, implementation, and challenges of the project.

### 3.1 Requirements

To begin improving our help request system, our team brainstormed what features we would need to have in our new system. First, we wanted to include forms to provide a structured way to field information from users and ensure that we were receiving all necessary information in their help request. Second, we wanted to be able to send users automated messages for multiple reasons, including reminding users to use the new features and redirecting users to other teams when we could not resolve their issues.

We also knew the implementation of a bot would be key in achieving a lot of the features we would need in our initial iteration and in future iterations. The main functionality of the bot we focused on for the MVP was the ability to support slash commands. We wanted these slash commands to act as a sort of search bar for our service so users could quickly retrieve information about the login service without having to sift through our documentation.

### 3.2 Design

Some of the features of our new help request workflow would require minimal programming as they could be implemented through Slack's Workflow Builder. Integrating a bot into our channel was a much

more involved process so our team decided to dedicate one of our weekly design meetings to discuss how to approach this. Since some of the developers on the team had some previous experience deploying similar applications, they provided some valuable insight on how to structure the architecture of this project.

To align with company-wide coding standards, we decided our bot would be fully backed by AWS resources. After multiple iterations in the design process, we decided on a simple structure, illustrated in Figure 1 below.

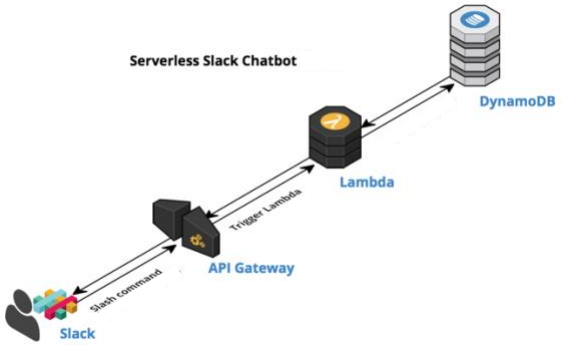


Figure 1: Diagram of Backend Architecture (Thorn Tech Staff, 2017)

First the slash command is initialized by the user in a Slack message. When that message is sent, Slack sends a HTTP POST request to our API Gateway. The API Gateway triggers our Lambda handler when it receives a POST request. The Lambda is responsible for verifying and parsing the request, and then pulling data from our DynamoDB table which houses the information we want to send back to the user. Once that data is retrieved, the Lambda buttons it up into a human-readable message which is sent back to the user. We settled on this simple architecture for the first iteration of the bot as it allows us to easily make changes in future iterations.

### 3.3 Implementation

To aid in the simplicity of this application, we decided it would be best to deploy our bot using AWS Cloud Development Kit (CDK), a way to specify our AWS infrastructure as code and deploy it with CloudFormation. My development team provided a boilerplate CDK application to begin the programming process. Then it was just a matter of adding the right resources, making sure everything was configured correctly, and writing the code for the Lambda handler function. During testing, we deployed the bot to our company's sandbox AWS environment. This ensured that if anything went awry in our application that it would not affect the other services. Currently, the bot lives in the alpha environment, which is still a testing environment but slightly more like the configurations in the production environment.

### 3.4 Challenges

One of the biggest challenges of this project was ensuring the security of the bot as this was a new concept to me. Since the bot had its own API endpoint, we needed to make sure it was secure and we would only receive requests from our Slack bot and nothing else. Luckily, Slack signs its requests using a secret that is unique to each bot. This made the request verification process much more stress-free. The process starts when a request is sent by Slack. First, we want to manually compute the signature by concatenating the version number, timestamp on the request, and the request body together and then calculate the SHA256 hash of that string with the unique secret as the key. Then we want to compare the computed signature with the signature contained in the request body. If these signatures match, then we can carry on with the rest of the process; otherwise, we abort the process.

## 4. RESULTS

The new help request workflow was rolled out in pieces as they became functional. The first parts of the new help request workflow made public on our Slack channel were the forms to field new help requests. They had an almost immediate impact, as all the developers on the team praised how necessary information was fed input at the inception of a request and there was no need to hound users for more details. The second part we decided to publicize was the automated reminder messages. We noticed that many people were not using the new features in the channel because they simply did not know the features existed.

These first two features came out quickly and both users and developers were very pleased with the outcome of their addition to the channel. We noticed much shorter message threads when help requests were filed, fewer help requests in our channel that were questions other teams handled, and a Slack channel that was much more pleasing to the eye. The beefier part of the help request workflow overhaul was the bot, which took quite a bit more time to roll out to our channel because of its complexity. We were not able to get a full version of the bot integrated into the channel due to time constraints, but we believe that the addition of the bot and its ability to automate trivial tasks could only add value.

## 5. CONCLUSION

The process of streamlining our help request system seemed like a daunting task when we began the project, but we quickly learned it was much simpler than we expected. Slack's expansive suite of automation tools and AWS CDK were the main actors in simplifying the development process. The project was very personally fulfilling as I walked away from a 10-week internship feeling like I made a significant difference for my team. Developing this project also exposed me to a lot of new techniques and technologies while also reinforcing the skills I already have.

## 6. FUTURE WORK

There are a few improvements that could be made to our new help request system that we weren't able to accomplish. First, the Slack bot should be deployed to the production environment and made public on our channel. We believe that the bot is the most valuable part of the new system, and we were only able to do some internal testing within our development team so making it public on our channel is our top priority. Once deployed, we want to expand the functionality of our bot by adding different slash commands and integrating it with our other development tools like Jira and BitBucket. Another improvement that could be made is how our bot retrieves information for the slash command that is currently implemented. Right now, our bot pulls information statically stored in our DynamoDB instance, but ideally it would pull from a more dynamic source as the information is constantly changing. This would be a much more involved process and we would likely need to restructure the backend architecture of the bot.

## REFERENCES

[1] Haughey, M. 2019. Build a self-serve IT help desk in Slack. (July 2019). Retrieved from https://slack.com/blog/productivity/build-a-self-serve-it-help-desk-in-slack

[2] Pilvelis, T. 2020. How to Create a Slack Bot using AWS Lambda in < 1 Hour. (Jan 2020). Retrieved from https://medium.com/glasswall-engineering/how-to-create-a-slack-bot-using-aws-lambda-in-1-hour-1dbc1b6f021c

[3] Thorn Tech Staff. 2017. How to Build a Serverless Slack Chatbot. (Mar 2017).

Retrieved from https://thorntech.com/serverless-slack-chatbot