Working at a Start-Up: Experiences and Learnings as a Data Science Intern

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

Ansel Sanchez

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

Brianna Morrison, Department of Computer Science

Working at a Start-Up: Experiences and Learnings as a Data Science Intern

CS4991 Capstone Report, 2022

Ansel Sanchez
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
acs7fge@virginia.edu

ABSTRACT

A local Charlottesville start-up, Babylon Micro-Farms, which specializes in developing smart cabinets for automated hydroponic produce farming, was rapidly growing and needed to innovate its produce monitoring system. As a data science intern, I was tasked with building a new dashboard of sensor metrics that would be easy to use and be highly presentable both to engineers and potential investors. Working in an Agile environment to complete the project, I used languages and technologies I was familiar with such as Python, Linux, and version control on top of a myriad of new tools learned throughout the internship experience. In the end, I developed, for both internal and client use, a full-stack data scraping, visualizing, and alerting application to fully monitor all of the company's farms. Although the application proved to be a success and a step in the right direction for a better data visualization and monitoring platform, improvements would be needed in the complexity of visualizing sensor data trends over time as well as in allowing for a better alerting system with company communication channels like Slack.

1. INTRODUCTION

In creating a start-up, attracting investors is of utmost importance in the early stages of the company. When I joined Babylon Micro-Farms (Babylon), they were in their early investment funding rounds, specifically series B, in need of growing the company even more. The company outlook was positive, but the co-founders had a greater vision for the company, which required more capital. They wanted to expand the movement of hydroponic smart cabinets across state lines and improve upon the automated technology they currently had. To do this, they needed to attract investors with a "wow" factor, something that would impress businesspeople and lure them in to invest in the growing company. That is why I, along with another intern, were brought in and tasked with developing a new dashboard of metrics able to monitor the data coming in from the smart farming cabinets, while showing aesthetically pleasing graphical representations of the metrics to be used by clients and engineers. Furthermore, and most importantly, with a system like this, the company would appear to be more serious, advanced, and sophisticated, which would be eye-candy to potential investors.

2. BACKGROUND

The company, Babylon Micro-Farms, was officially founded in 2017 at the University of Virginia by Alexander Olesen and Graham Smith. The pair were already working on a prototype of their smart farming cabinets within the i.Lab at UVA, an incubator that fosters support for students trying to transform ideas into products and businesses. Babylon's main product is its vertical, hydroponic smart farms that grow produce within a software-controlled cabinet.

Hydroponic farming is an alternative to traditional farming techniques in that plants are grown without soil and instead are set in more nutrient-rich materials and water. Hydroponic farming boasts a myriad of benefits over traditional farming, including higher crop yield, less farming space, less water usage, increased feasibility during all seasons, and reduced supply chain issues. Babylon's goal is to elevate hydroponic farming by creating a turnkey farming cabinet for businesses and everyday people that automates the growing and harvesting process through software and artificial intelligence (AI).

Their key product is the Galleri, a smart-farming cabinet that can grow more than 45 plant varieties. The cabinet contains various sensors that

control the growing conditions including water level, pH level, humidity, light intensity, and electrical conductivity. These Internet of Things (IoT) sensors are programmed using AI and proprietary Babylon software to allow for the perfect growing conditions for all of their different crops and for varying growing situations. At the time I joined the company in 2020, the only "easy" way for the engineers to look up farm data from these sensors was via their app. They needed a more central and sophisticated dashboard of this data that could perform necessary alerts to engineers when needed.

3. RELATED WORKS

In developing a central dashboard of metrics of incoming IoT sensor data for all of Babylon's farms, there really was not previous or example work for the other intern and me to base our application on. We were given the freedom in designing the UI, including how the data would be displayed and how alerts would be sent to engineers. We did, however, use tutorials for a lot of the platforms and technologies that the application was built with. For example, we were unfamiliar with Prometheus, the data scraping component of the application, and Grafana, the visualization component of the application, and so we followed their "Getting Started" tutorials to show us how each technology could be leveraged [1][2]. And although these tutorials did not give us complete inspiration in the design and development of the full system, they did give us a sound foundation on how to use these technologies to successfully incorporate them into the final application.

4. PROJECT DESIGN AND DEVELOPMENT

The application was separated into three key microservices: data monitoring, data visualization, and status alerting. Each was contained in its own Docker Container, a unit that packages software and makes it easy to deploy by itself or with other Docker Containers. Although the application was separated into respective functionalities and services, the whole application was able to spin up all the services at once with a simple Docker command inputted via the command line terminal. The separation of each microservice allowed for ease of development, as changes and refactoring to one Container's code would not affect the functionality of the other Containers. YAML files were written to configure how each Container would be spun-up, including which files would be ran and what endpoints various webpages were hosted at.

4.1 Data Monitoring

The basis for the data monitoring microservice was in utilizing a data monitoring platform called Prometheus, a software that allows engineers to easily pull data from databases or other data storage options. In my case, the data was hosted on a Flask MQTT webpage, which essentially is a webpage built with Python that utilizes the MQ Telemetry Transport (MQTT) protocol to retrieve data from the IoT sensors in each of the smart farming cabinets. This webpage that held all the data and had been built prior to me joining Babylon and provided an easy way for me to configure Prometheus to be able to scrape this data, as all the data was held together in one easily accessible source and in a standard format. Each line of the webpage held a sensor data value for one farm along with metadata such as the sensor type, timestamp, and farm ID. However, some modifications were made in how the actual data was formatted on the webpage.

Since the goal of the entire application was to make a dashboard of metrics that made it easier to read and assess the health of each of their farms, the actual farm name would be beneficial to have attached to each line of data so that engineers would not need to remember each farm ID. To fix this issue, I edited the Python program that created the webpage to be able to match each farm ID to its actual farm name and append it to its corresponding data lines on the webpage. To allow Prometheus to scrape this data, I created numerous configuration files that were written in the YAML data-serialization language to be able to tell Prometheus at which endpoint to look for the data and in what ways to pattern match to pull and group various data together.

For example, when the webpage that hosts the data is running, it exposes itself at a certain endpoint in order for systems to connect and see it. So, I wrote YAML files that would spin up a separate Prometheus webpage that would connect to the data webpage's endpoint and designed pattern matching code that would group together data for each specific farm. So, for example, if one were wanting to see all the sensor data for a farm with the ID of "1", the user would use the query field and type in that specific farm ID and all the sensor data for that farm would appear in a list or if one would want to see all pH value data for all farms,

one would query “pH” on the Prometheus webpage and all pH values for all the farms would appear in a list.

4.2 Data Visualization

Once the data monitoring part was completed, we needed a way to display this data aesthetically. Grafana, a third-party data visualizing platform, was used to achieve this due to its ability to connect easily with data scraped using Prometheus. YAML files were written to be able to spin up a Grafana webpage that would connect to the Prometheus webpage exposed at a certain endpoint. Once the Grafana and Prometheus webpages were connected, the configuration and design of the graphs and visualizations of the data were done all within the Grafana webpage’s UI.

The other intern and I had creative freedom in the types of graphs to display, how the dashboard should look, and what sensor data to visualize. We decided that since this would be a dashboard of metrics for each farm, all sensor data would be relevant to display. We first designed and created line graphs to model data over time. These sensor values included water level, electrical conductivity, pH, and flow rate. These were all displayed on one graph, and one could filter out which data were to be displayed on the graph. Users could choose to see all the line data or choose to show a select few with each other. There were also binary data sensor values that included whether the light is on, the water pump is on, and if there was irrigation occurring. These were displayed separately with each sensor value occupying a card-style block that displays whether the value is “On” or “Off” and would be displayed as green if it’s on and red it it’s off. An example of the visualization can be seen in Figure 1. Lastly, the temperature sensor value would be displayed in a similar fashion to the binary variables, as it occupied its own square and displayed the temperature on this card-style block. Once these visualizations were complete, users would be able to query for a specific farm on the Grafana webpage and it would display all relevant sensor data for that queried farm.

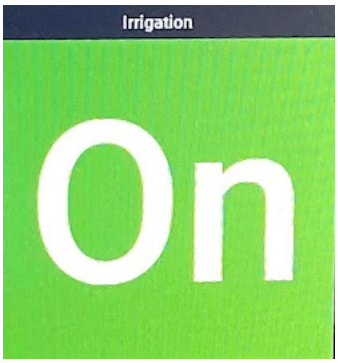

Figure 1: Card-style block displaying binary sensor data

4.3 Status Alerting

The last component of the application is the status alerting feature. Prometheus can be configured to send out alerts if the data it scrapes goes over a certain threshold set by the configurer. The other intern and I discussed with the senior engineers to get a sense of what kind of values would raise alarm to them if data went above or below a certain threshold. For many of the timeseries data, certain threshold levels were discussed and were needed to be implemented into the application to send alerts to engineers. YAML files were written to set up what sensor values should check for these threshold values as well as how the notifications would be set up. As this was just the initial iteration of the application, alerts were sent via email to the engineers. These alerts contained the farm name, farm ID, sensor value, and timestamp.

5. RESULTS

At the conclusion of my internship, a basic full-stack application that monitored, displayed, and alerted engineers of data from each of their hydroponic farms was completed. The application is currently being used by Babylon as a way to keep track of the health of their farms and as a way to “wow” potential investors by illustrating a more sophisticated oversight system of their farms.

Prior to the competition of the system, engineers had a hard time looking at all their sensor data at once. Now, engineers can quickly lookup farms, either all at once or query them individually, to have better visibility on the activity and health of their farms’ sensor data. With this application in use, Babylon has been able to grow significantly, as it attracts the capital of investors as well as makes them

more efficient by having easy access to their sensor data in one application.

6. CONCLUSION

At the end of my internship with Babylon Micro-Farms, a full-stack data monitoring, visualizing, and alerting system was developed. This dashboard of metrics served as an all-encompassing application for engineers to gain better visibility of their farm data in an easy-to-use way. This also allowed for better data monitoring through visually appealing graphs and notifications for concerning statuses of farm metrics. Furthermore, this helped to greatly increase capital for the start-up, as it allowed Babylon to show investors a more sophisticated way of tracking their data, which gives confidence to investors that the company is serious in their goal of creating a more sustainable future.

7. FUTURE WORK

For further development of this project, it would be more beneficial to the engineers to add in more complex graphs that can visualize different data trends together. This would allow engineers to be able to diagnose problems with more information by being able to see patterns amongst data trends that might be similar to each other. Furthermore, the notification system should also be improved. Currently, notifications are sent via email. However, most of the engineers at Babylon use Slack as the primary communication channel, so an integration of the application's notifications to be sent to Slack would be ideal.

8. ACKNOWLEDGMENTS

I would like to thank my partner, Deniz Cakmak, who was the other intern during my summer internship with Babylon Micro-Farms. She played a crucial role in helping to develop this application. I'd also like to thank the lead engineer that worked as our supervisor, Amandeep Ratte. He provided valuable support in trying to help us solve problems we had, provide us with materials and tutorials for success, and general guidance to lead us in creating the final deliverable.

REFERENCES

[1] Grafana. Get started with Grafana and prometheus: Grafana documentation. Retrieved October 21, 2022 from https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/

[2] Prometheus. Getting started: Prometheus. Retrieved October 21, 2022 from https://prometheus.io/docs/prometheus/latest/getting_started/