

Background

- Tufts Technology Services (TTS) handles technology solutions for the entirety of Tufts campus, including wifi networks and many on-campus devices.
- TTS has traditionally used enterprise services for alerts and monitoring in relation to their user logs.
 - Premium Elasticsearch services, specifically Elastic Alerting.
- Within the past 2 years, TTS has changed to a free license and as a result has lost its previous alerting capability.
 - Open source projects exist, but open source had supportability issues with Kibana and Elasticsearch.
- Effective and efficient alerting is a core part of how TTS successfully maintains a proper security posture.
- Without such alerting features, it is easy for potential issues to go unnoticed.

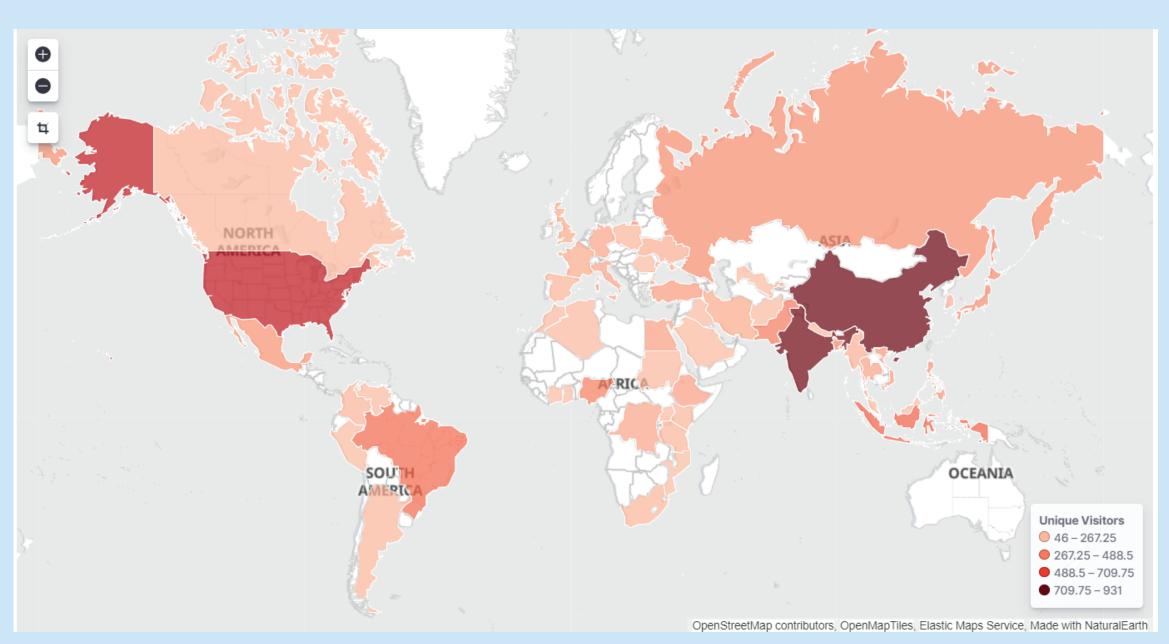


Figure 1: Unique user heatmap for TTS network over the course of a month, showing both the variety of user of the TTS network and the scale of the information that TTS logs store.

Goal

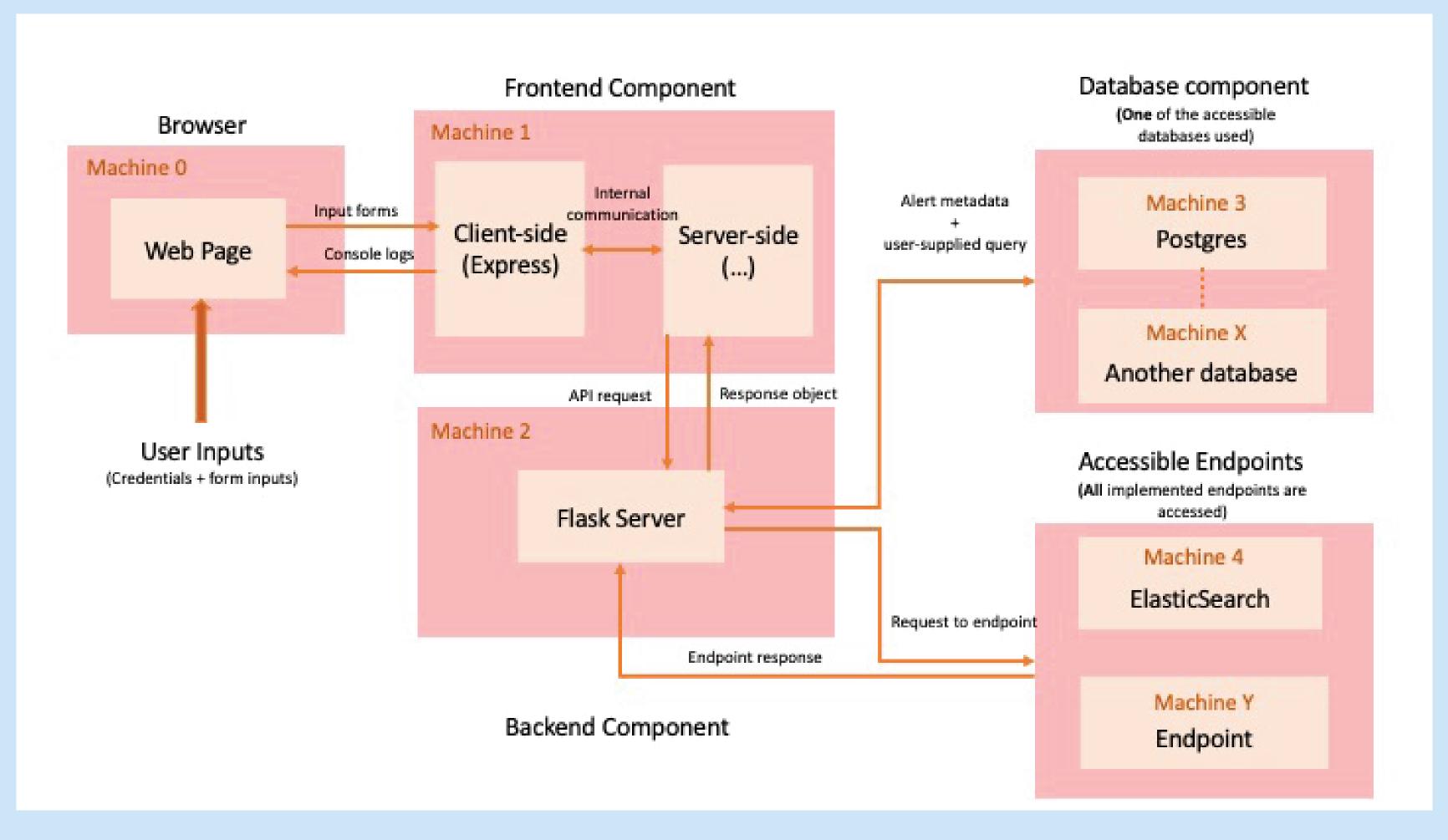
To design and develop an alert system that interfaces with TTS's ElasticSearch clusters; the alert system will have a user-friendly frontend, a moddable backend, and will incorporate standard enterprise security features.

Observations

- Implemented Features
 - Input format by query type
 - Single Sign-On (SSO) metadata hosting
 - Internal alert manager
 - HTTPS hosting
- Next Steps
 - SSO integration
 - Alert histories
 - Flask server in production mode
 - "Graceful" start, stop, and restart of backend server

TTS NextGen Alerts





Reflection and Conclusion

Industry Takeaways

muusiiy laneaways			
 Security concerns are a lot more 	• The s		
serious on the enterprise level.	requi		
 Previous projects, even within 	know		
webdev, did not prepare us for real	unde		
security requirements.	speci		
 Scalability is needed for data and usage 			
of this magnitude.	goals		
 Many design decisions were made 	Modu		
around supporting multiple users at me			
once or speed optimizations.			
 Work needs to be presented to 	٥ Th		
non-technical people when working	mc		
with an organization.	COI		
 Getting our descriptions high-level 	- Know		
enough to be useful took time.	was		
	influe		
	• We		
	fro		
	wh		
	the		
Techno	logio		
	lugies		
expr	229		
	000		
python Flask			
PostgreSQL	Ke		

Project Takeaways

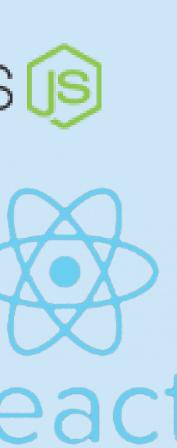
scale and scope of our project uired professional enterprise-level vledge and comprehensive rstanding of non-technical ifications. We were unprepared for r and as a result did not meet our

ular design allows for easier ification and extensibility for future lopment.

he API of the flask server's internal odules will remain relatively nsistent.

wing which features to prioritize difficult and relied on multiple ences.

Ve had to juggle immediate asks om our sponsors while considering hat order of implementation made e most sense.





Advant

- Modular design for code reuse.
- Lots of support documentation.
- Easy integration Firebase.

Figur

Advant

- Easy SSO integ
- Lightweight, us HTML.

Figure 3: Expr UI demo

Advantages

- Large collection of usable libraries.
- Deep knowledge within group.

Figure 4: Backend demo

James Cameron, CS Kate Hanson, CS Muhammad Umair, CS Andrew Wang, CS

Sponsor: Tufts Technology Services

	D	esign	
	sted on	ntend: React Node server Disadvantages • Not easily SSO-compatible.	
ting n witl	h	 <u>Landing</u> <u>Home</u> <u>Account</u> Sign Out Tabs Demo	
re 2: React I demo		Multi-Bool Query Filter Query Aggregate Query History Start Time: Now St	
Fin	al Fron	Match to Value: is + Submit tend: Express	
Hosted on Node server ages Disadvantages ration. • Less modular in it's design. ses base			
ess ^R	NextGen Multiple Boolean Caw Juery:	Alerts Filter Aggregate Raw	

Backend: Flask Server written in Python Disadvantages Difficult to convert to production.

• Elasticsearch included.

Submit

🗖 👝 📄 novtgonalarting, backand 🖉 mumair01@infaces day, 02. /n
💿 😑 📄 nextgenalerting_backend — mumair01@infosec-dev-03:~/n
<pre>/opt/py_venv/test_env/lib64/python3.6/site-packages/elasticsearch/connection/htt p_urllib3.py:154: UserWarning: When using `ssl_context`, all other SSL related k wargs are ignored</pre>
"When using `ssl_context`, all other SSL related kwargs are ignored"
* Serving Flask app "Development server" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployme
nt.
Use a production WSGI server instead.
* Debug mode: on
* Running on https://10.246.104.129:5000/ (Press CTRL+C to quit)
10.246.104.129 [08/Apr/2021 10:43:37] "POST //connect HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:43:37] "POST //operation HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:43:37] "POST //operation HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:43:37] "POST //operation HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:43:37] "POST //operation HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:43:37] "POST //disconnect HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:44:02] "POST //connect HTTP/1.1" 200 -
10.246.104.129 - [08/Apr/2021 10:44:02] "POST //operation HTTP/1.1" 200 -
10.246.104.129 - [[08/Apr/2021 10:44:02] "POST //operation HTTP/1.1" 200 on
10.246.104.129 [08/Apr/2021 10:44:02] "POST //operation HTTP/1.1" 200 -
10.246.104.129 [08/Apr/2021 10:44:02] "POST //disconnect HTTP/1.1" 200 -
Emoil cont