[Team Members](#_2et92p0)

[Part 1 - Serverless BERT with Huggingface and AWS Lambda : Tutorial](#_r26inkestjp9)

[Part 2 - Limiting API Invocations from User](#_3j1oj13vquo)

[Part 3 - Build and deploy NER and Summarization APIs](#_fawmb2f9s6a7)

[Create a custom docker image](#_jkeez29x33gr)

[Deploy a custom docker image to ECR](#_ehrzpcmkn7z)

[Deploy AWS Lambda function with a custom docker image](#_9hz0lpxpfvv)

[Test the Serverless BERT API](#_8uyqx9k5ri43)

[Deployed and running NER API on AWS -](#_543hncsv94dk)

[Deployed and running Summarization API on AWS -](#_ga5nmqxaw61t)

[Updated Streamlit application that display/render the Summarization and Named Entity Recognition](#_cavdkkgypgfy)

[Updated Data Pipeline for SEVIR Nowcasting model [Media Use Case]](#_g9lgpxg75r8r)

[Part 4 - Build a LIVE admin dashboard](#_2y70yca637ag)

[Conclusion](#_2jxsxqh)

[References](#_z337ya)

[What’s next?](#_9mqifffit2ew)

[Further reading](#_lqephukhewqt)

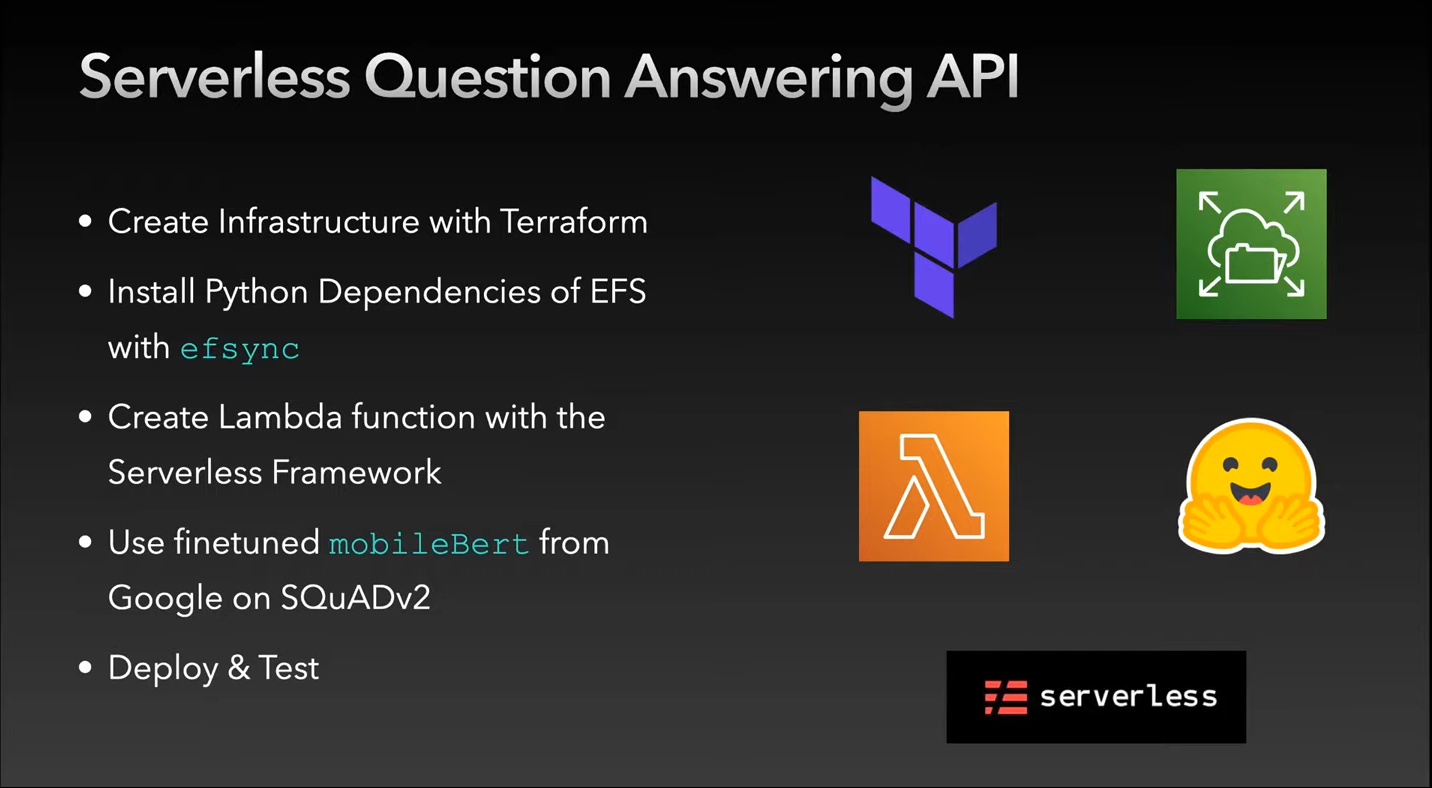
[Reference docs](#_an95hoqxrlhg)

# Team Members

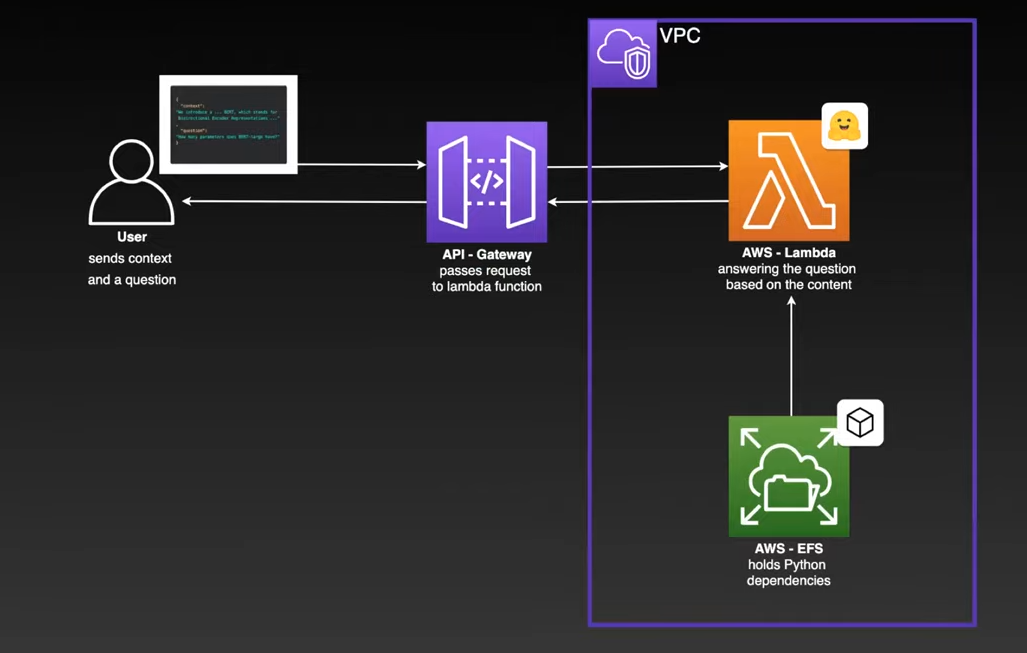
| NAME | NUID |
| --- | --- |
| PRIYANKA DILIP SHINDE | 001524484 |
| KSHITIJ ZUTSHI | 001021288 |
| DWITHIKA SHETTY | 002114630 |

# Part 1 - Serverless BERT with Huggingface and AWS Lambda : Tutorial

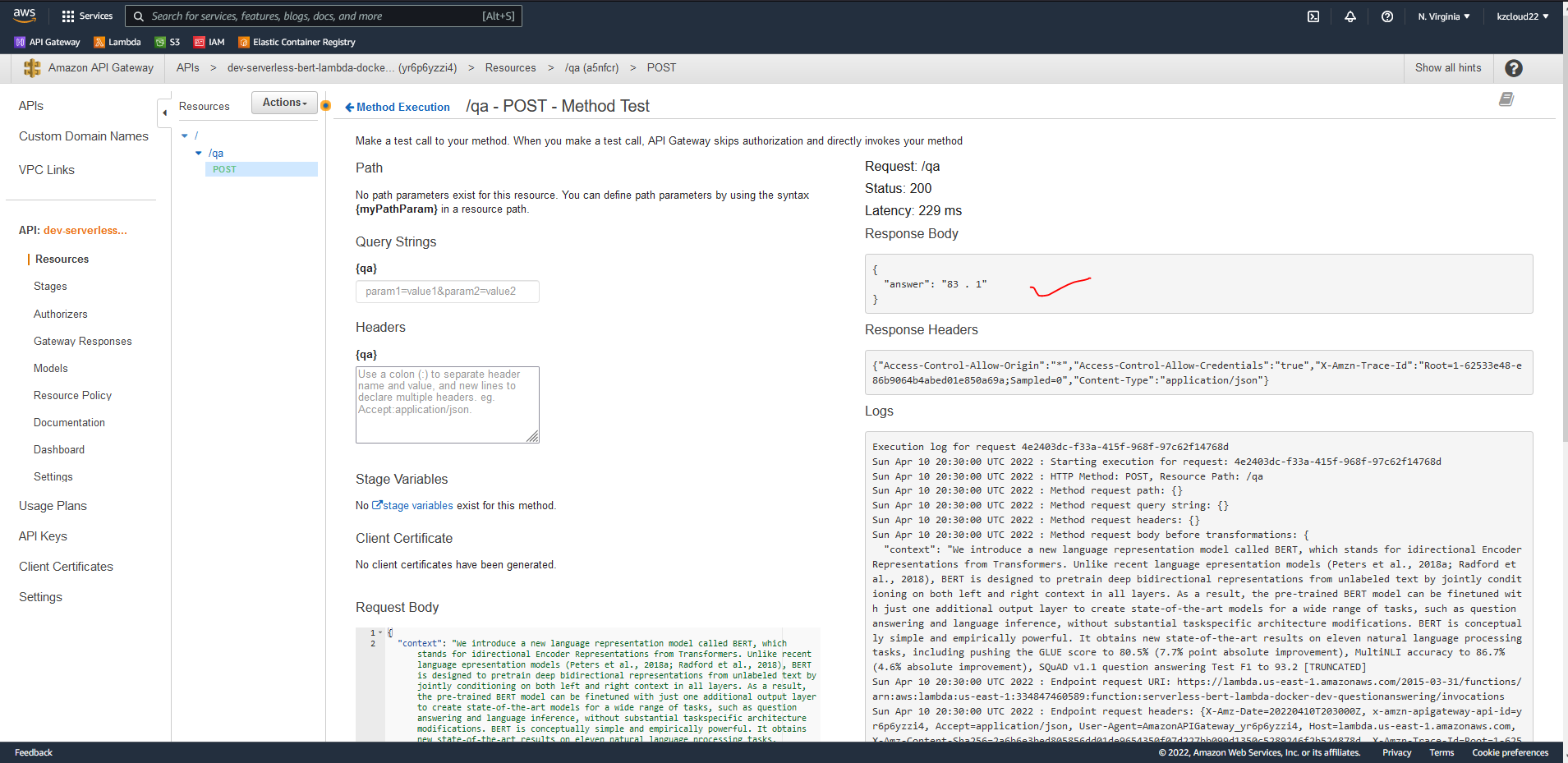
**Problem statement** -



The model architecture -



**Test the deployed Q&A API**

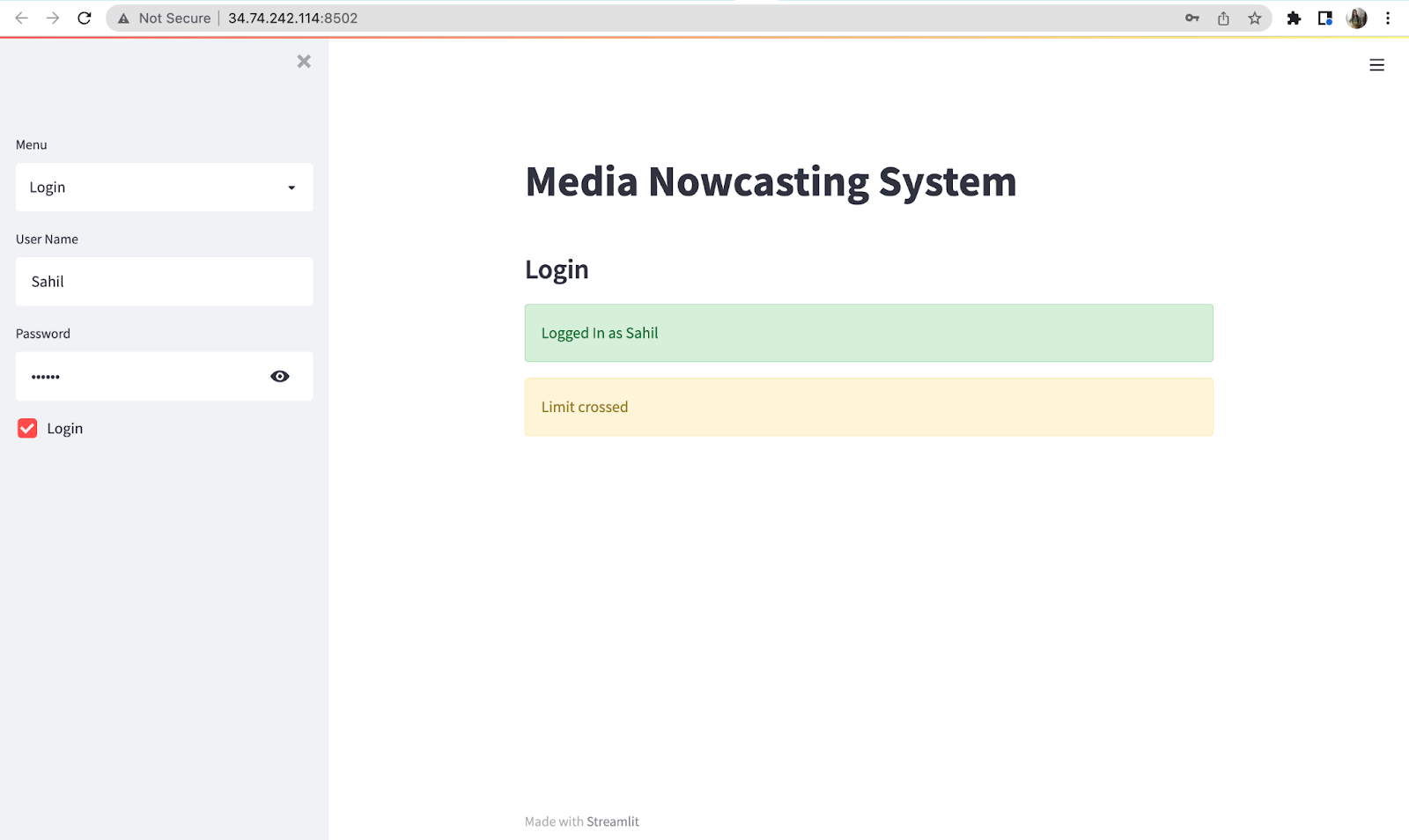
****

Complete code and description is available on Github - [Serverless-Question-Answering-API](https://github.com/kshitijzutshi/Serverless-Question-Answering-API)

# 

# Part 2 - Limiting API Invocations from User

Users can only use Nowcast Intelligent System for 10 searches after that it will throw an error **Limit crossed**.



# Part 3 - Build and deploy NER and Summarization APIs

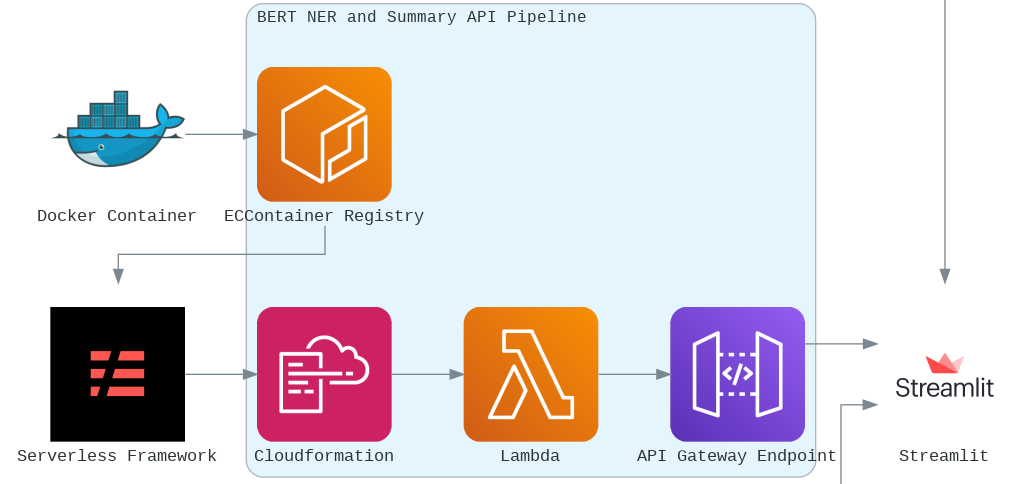
NLP has various implementations in a Data pipeline but particularly to gain insights through inference of the model, In our case of media nowcasting, we dockerized and deployed 2 NLP services namely - Named Entity recognition and Text Summarization. In order to facilitate the deployment of APIs we used the Serverless framework.

These two APIs will be performing operations on Event and Episode Narrative descriptions in the Storm events data for the user specified location.

For the BERT Named Entity Recognition API we used the following Hugging face model - [dslim/bert-base-NER](https://huggingface.co/dslim/bert-base-NER)

And for the BERT Summarization API we used the following Hugging face model - [mrm8488/t5-base-finetuned-summarize-news](https://huggingface.co/mrm8488/t5-base-finetuned-summarize-news)

The pipeline for which is as follows -



In build and deployment of both the APIs following steps were involved -

## Create a custom docker image

Before we can create our docker we need to create a requirements.txt file with all the dependencies we want to install in our docker.

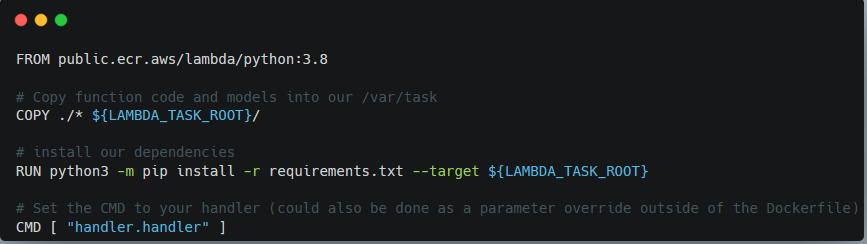
Requirements for the APIs -

https://download.pytorch.org/whl/cpu/torch-1.5.0%2Bcpu-cp38-cp38-linux\_x86\_64.whl

transformers==3.4.0

Building the docker file -

To containerize our Lambda Function, we create a dockerfile in the same directory and copy the following content.



To build our custom docker image we run - **docker build -t <docker-image-name> .**

## Deploy a custom docker image to ECR

Since we now have a local docker image we can deploy this to ECR. Therefore we need to create an ECR repository with the name <docker-image-name>.

**aws ecr create-repository --repository-name <custom-name>**

To be able to push our images we need to login to ECR. We are using the aws CLI v2.x. Therefore we need to define some environment variables to make deployment easier.

**aws\_region= eu-east-1**

**aws\_account\_id= <Your AWS Account ID>**

**aws ecr get-login-password \**

**--region $aws\_region \**

**| sudo docker login \**

**--username AWS \**

**--password-stdin $aws\_account\_id.dkr.ecr.$aws\_region.amazonaws.com**

Next we need to tag / rename our previously created image to an ECR format. The format for this is **{AccountID}.dkr.ecr.{region}.amazonaws.com/{repository-name}**

We can do it the following way -

**docker tag <docker-image-name> $aws\_account\_id.dkr.ecr.$aws\_region.amazonaws.com/<AWS-docker-image-name>**

To check if it worked we can run docker images and should see an image with our tag as name

Finally, we push the image to the ECR Registry.

**docker push $aws\_account\_id.dkr.ecr.$aws\_region.amazonaws.com/<AWS-docker-image-name>**

## Deploy AWS Lambda function with a custom docker image

In order to deploy our docker image, we are using the serverless framework that helps us orchestrate the cloud stack formation - which means it configures the lambda functions and API gateway endpoints for us, hence allowing us to get the final API url that can be used to make requests.

To use a docker image we need to have serverlss.yaml where we have the image and in our function section. The image has the URL to our docker image and also the digest value.

For an ECR image, the URL should look like this

**{AccountID}.dkr.ecr.{region}.amazonaws.com/{repository-name}@{digest}**

In order to deploy the function, we run **serverless deploy**.

**serverless deploy**

## Test the Serverless BERT API

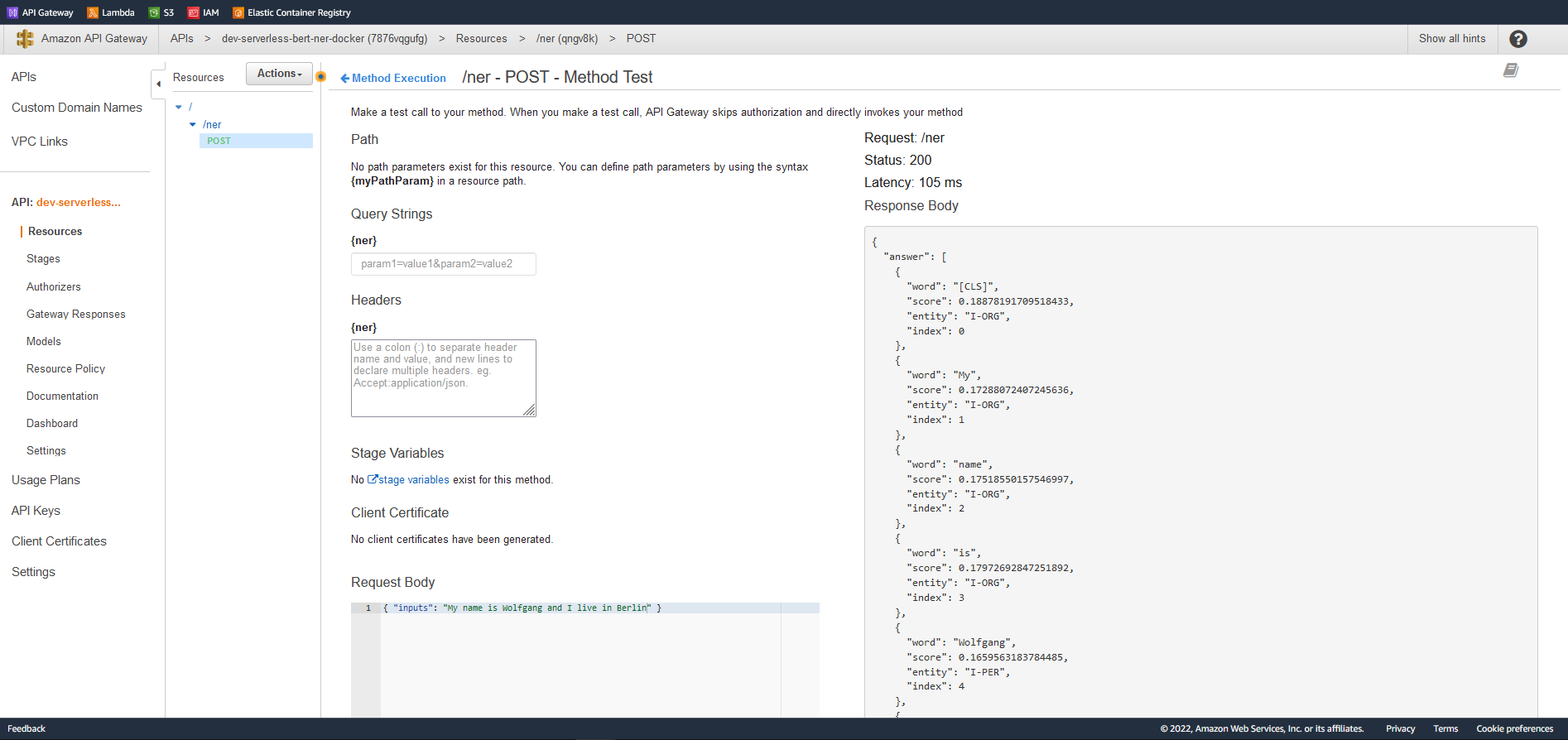
To test our Lambda function we can use Insomnia, Postman, or any other REST client. Just add a JSON with a context and a question to the body of your request.

*{*

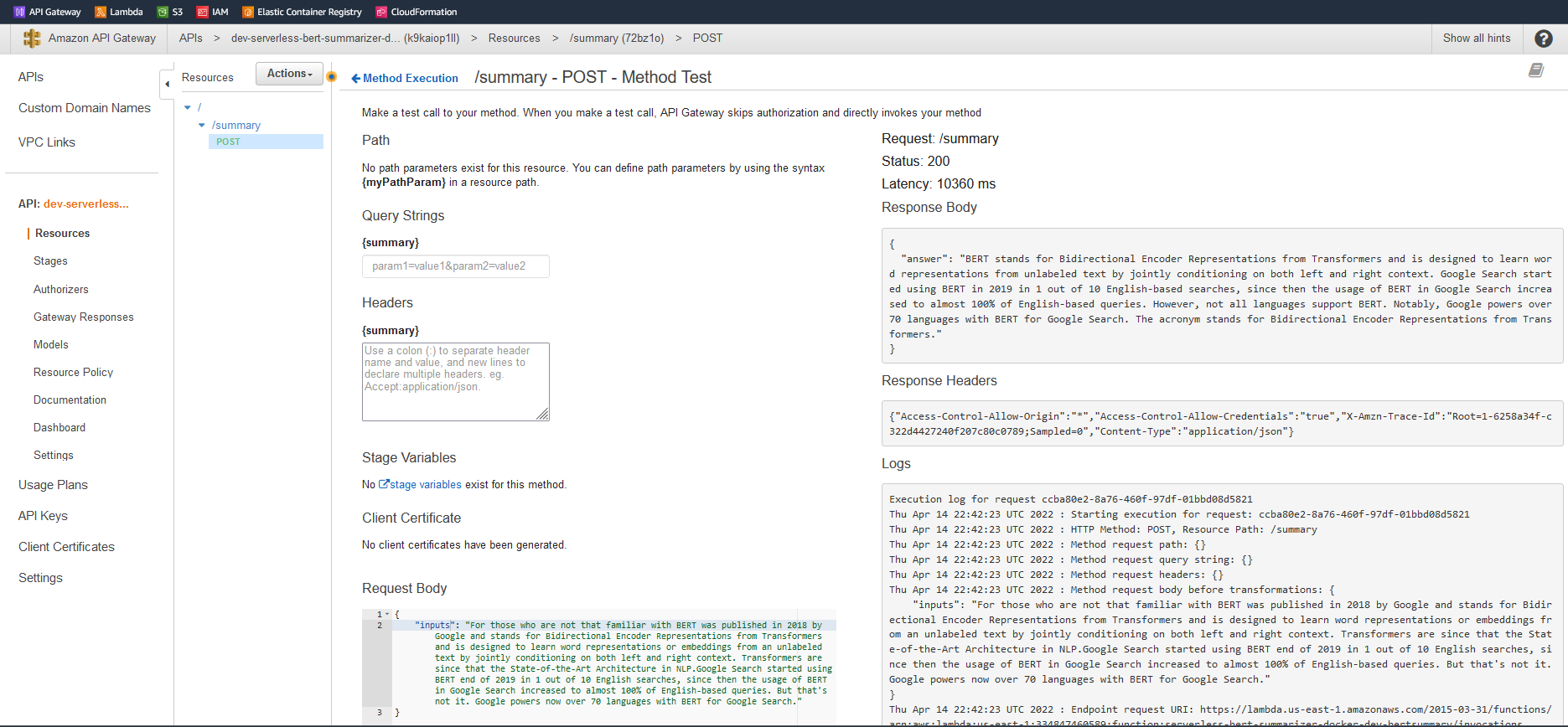
*"inputs": "We introduce a new language representation model called BERT, which stands for bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one."*

*}*

### Deployed and running NER API on AWS -



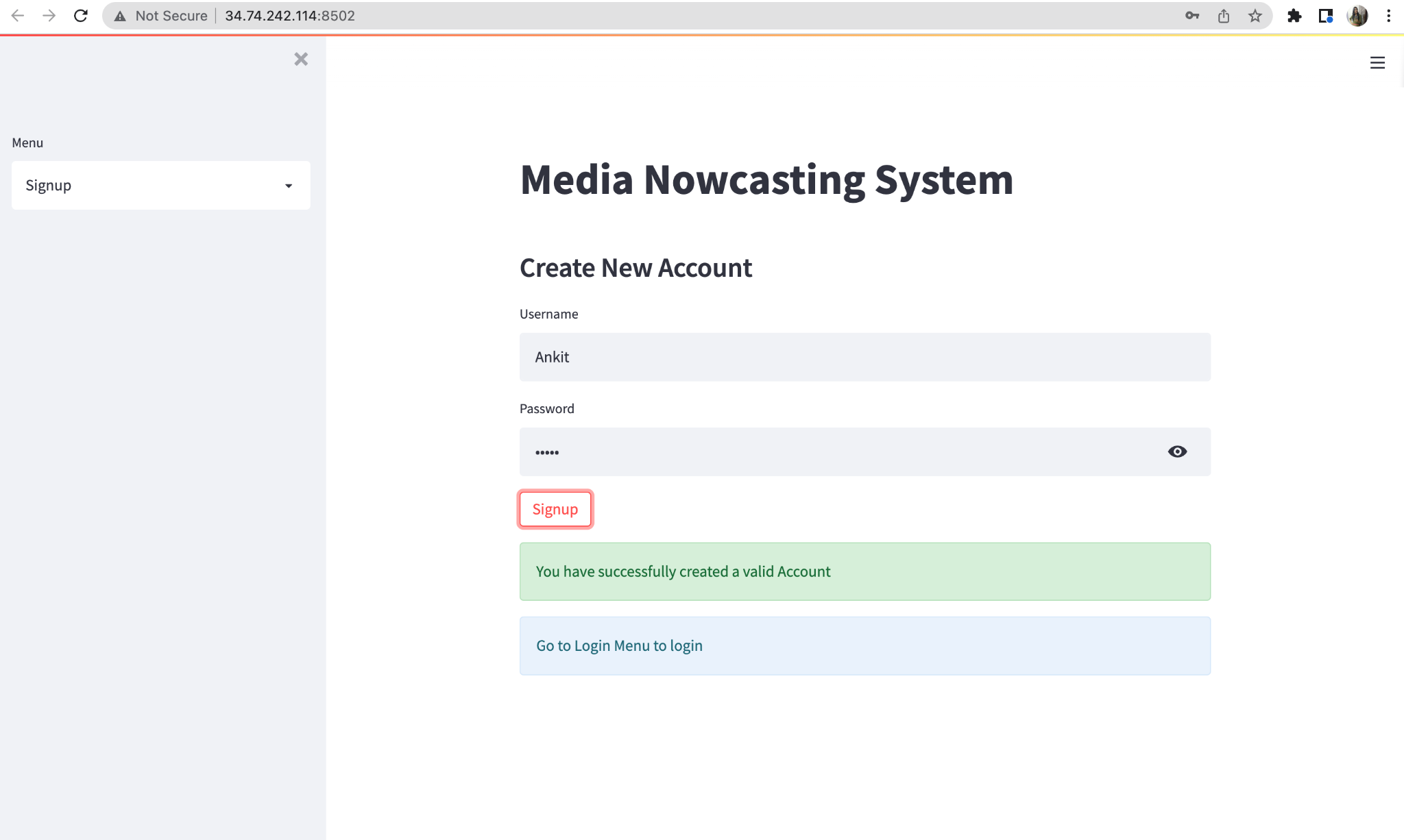
### Deployed and running Summarization API on AWS -

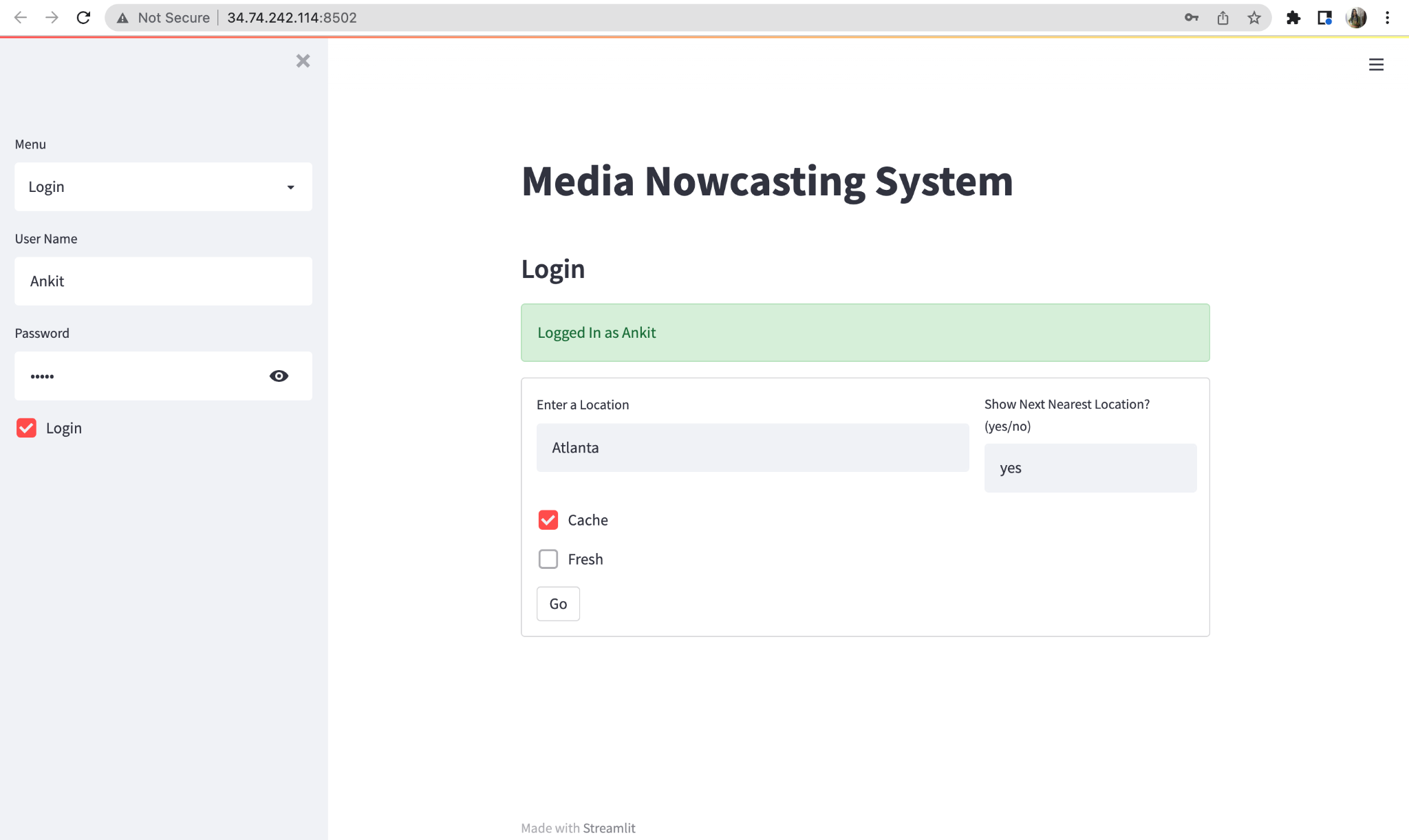


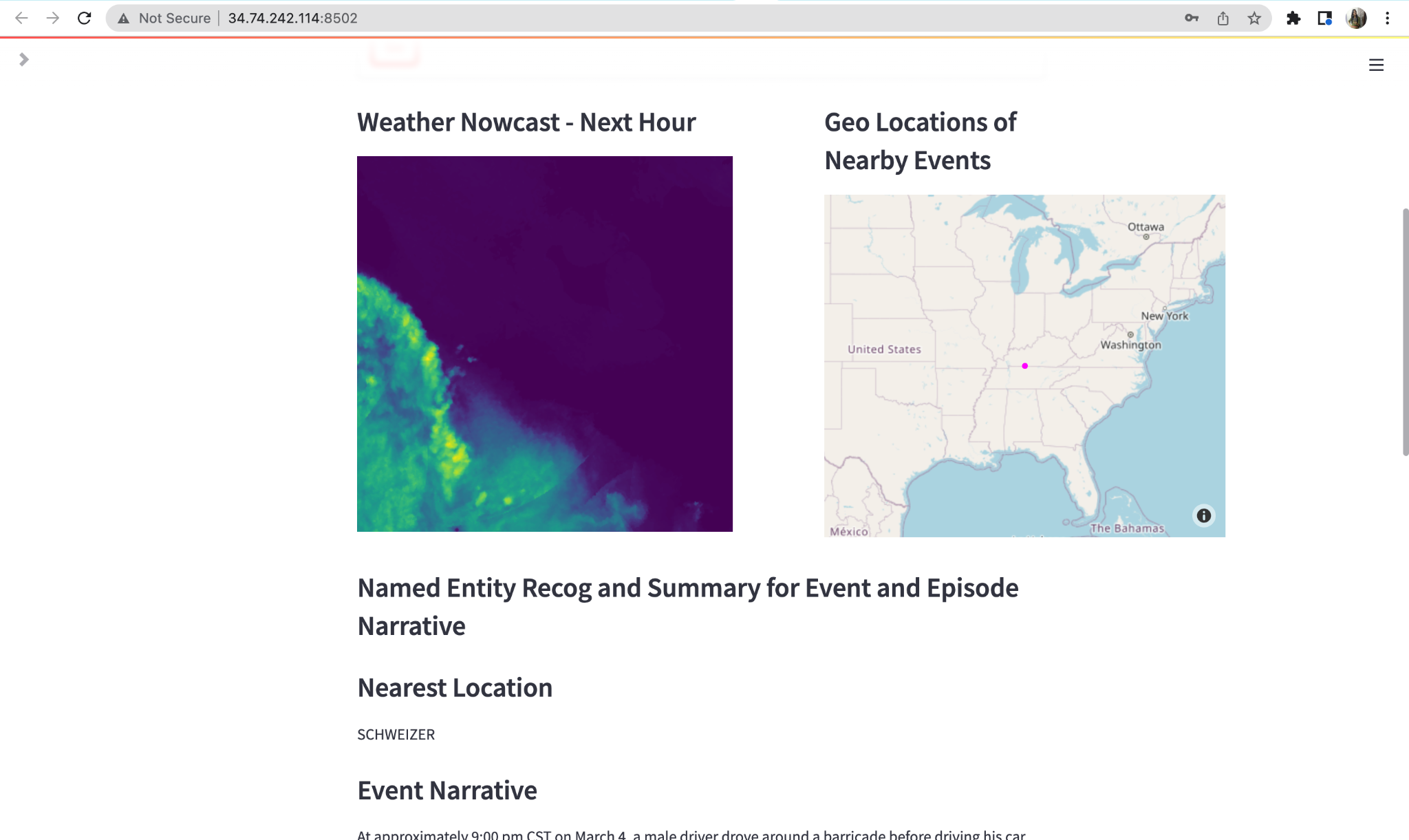
### Updated Streamlit application that display/render the Summarization and Named Entity Recognition

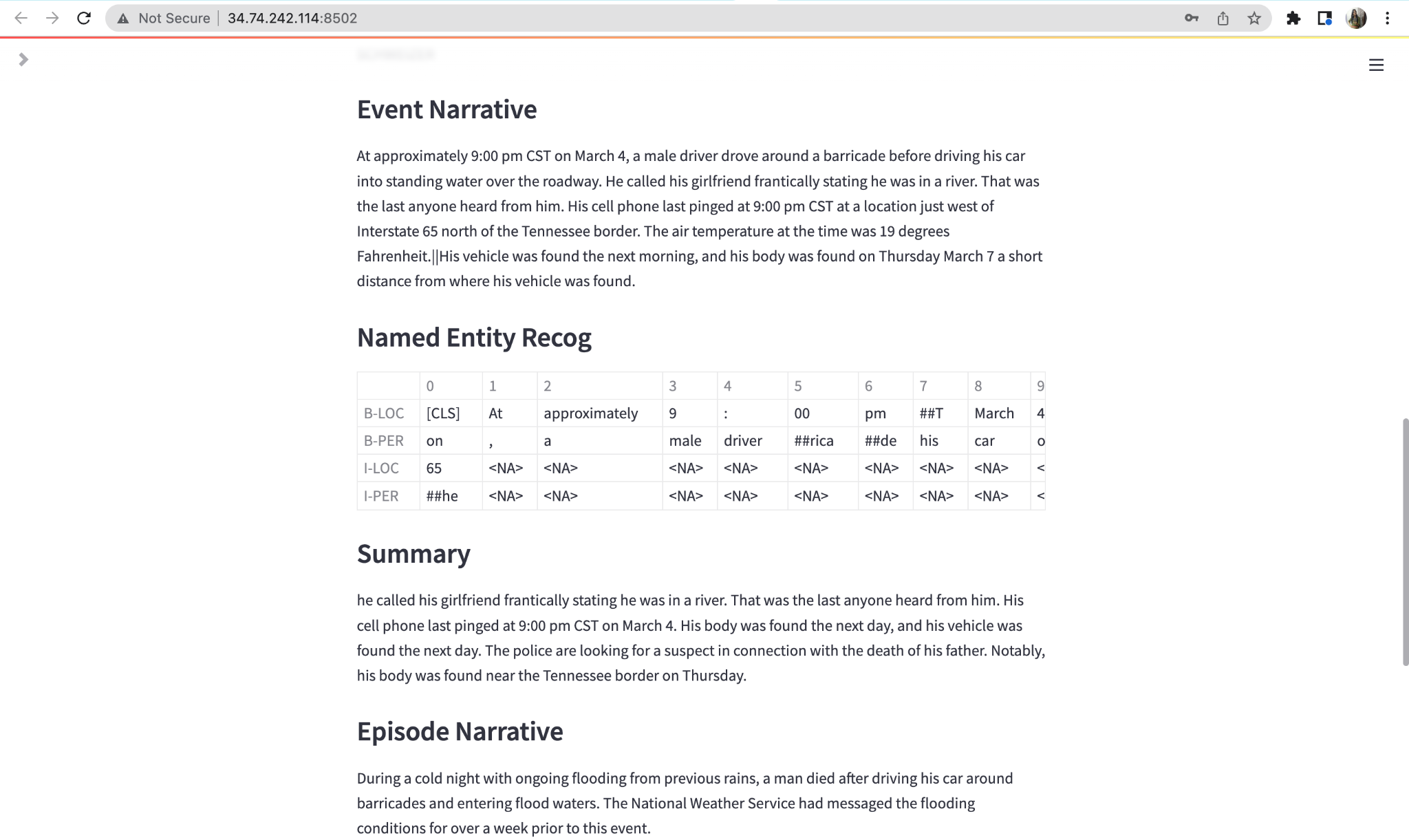
### 

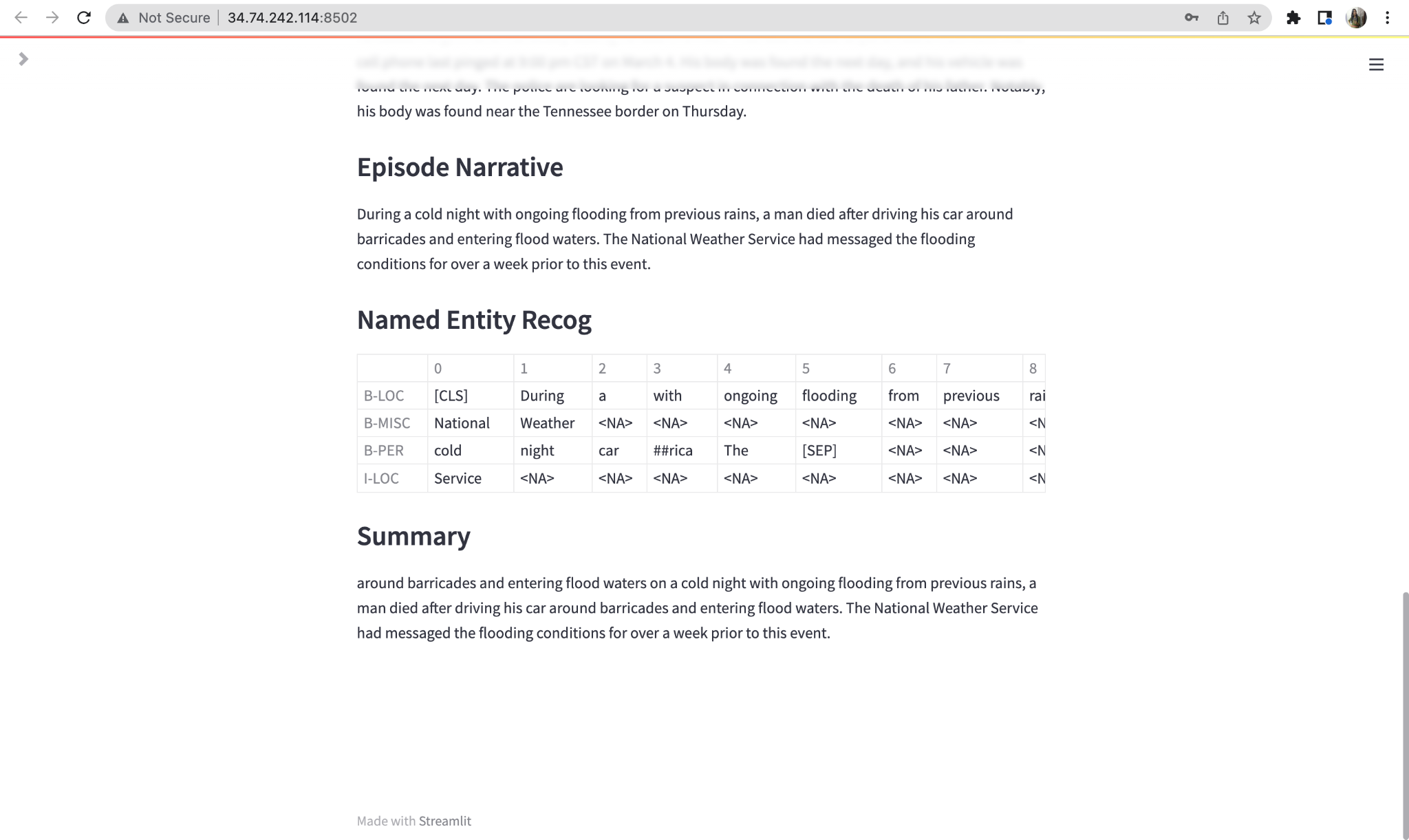
### 



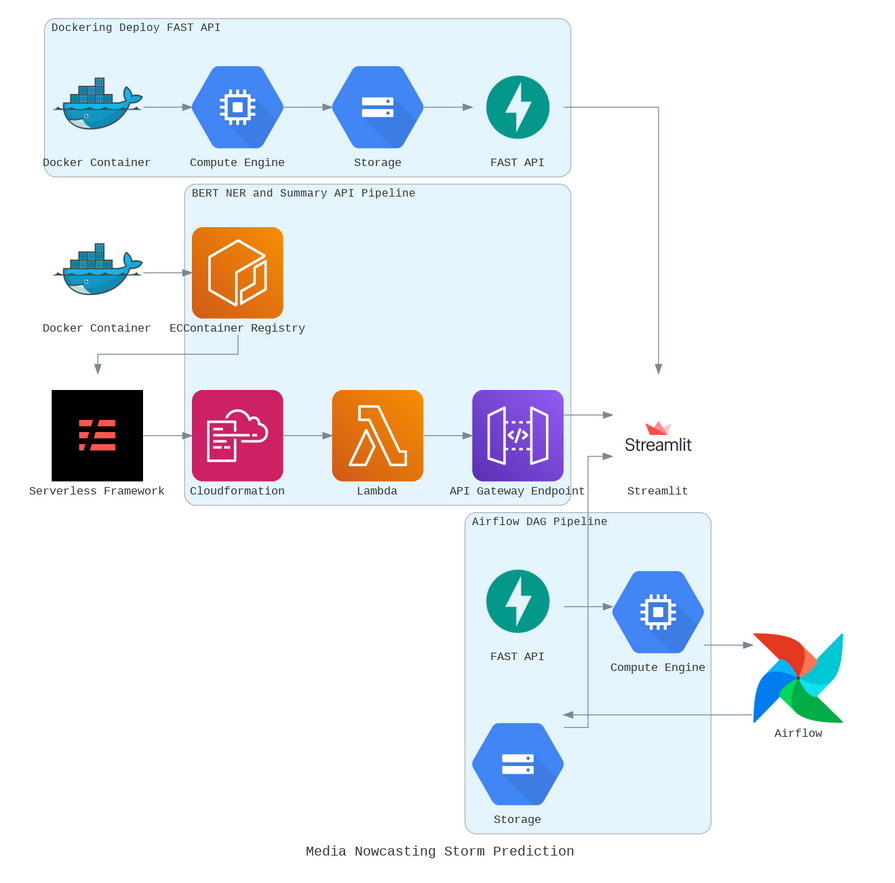








### Updated Data Pipeline for SEVIR Nowcasting model [Media Use Case]

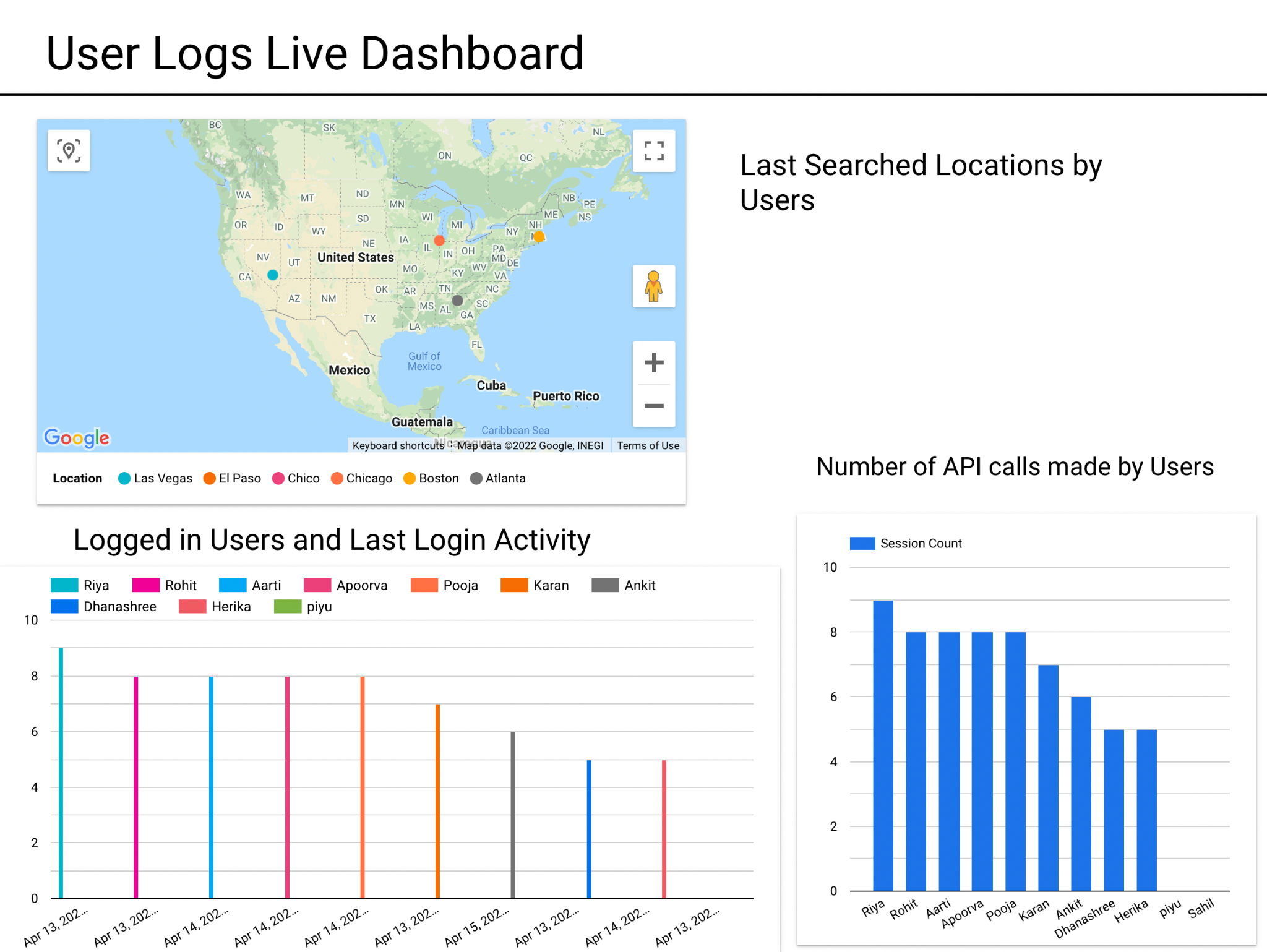


# Part 4 - Build a LIVE admin dashboard

We are exporting data directly from the GCP bucket which contains user log information to generate Live User Logs Dashboard in Google Data Studio.

The following Dashboard gives user usage information about:

1. When this service was last used by the user
2. Number of session counts left per user
3. For which location user is getting the nowcast generated



# Conclusion

1. Updated the SEVIR application so an authenticated user can only use unto N requests to the api.
2. Build a LIVE admin dashboard in GCP Data Studio that would reveal the user analytics in real time and how many queries each user has invoked.
3. Built two APIs: Summarization and NER using hugging face models. Both APIs are hosted and you will invoke them using the Sevir application for the summary and NER for the closest event and episode ID.

# References

1. <https://www.philschmid.de/serverless-bert-with-huggingface-aws-lambda-docker>
2. <https://nbviewer.org/github/MIT-AI-Accelerator/eie-sevir/blob/master/examples/SEVIR_Tutorial.ipynb>
3. <https://cloud.google.com/bigquery/docs/visualize-data-studio>
4. <https://huggingface.co/dslim/bert-base-NER>
5. <https://huggingface.co/mrm8488/t5-base-finetuned-summarize-news>
6. <https://www.youtube.com/watch?v=Q10vhymrGh4>

**“WE ATTEST THAT WE HAVEN’T USED ANY OTHER STUDENTS’ WORK IN OUR ASSIGNMENT AND ABIDE BY THE POLICIES LISTED IN THE STUDENT HANDBOOK”**