

Paige Rosynek

CSC 5201 301

Dr. Jay Urbain

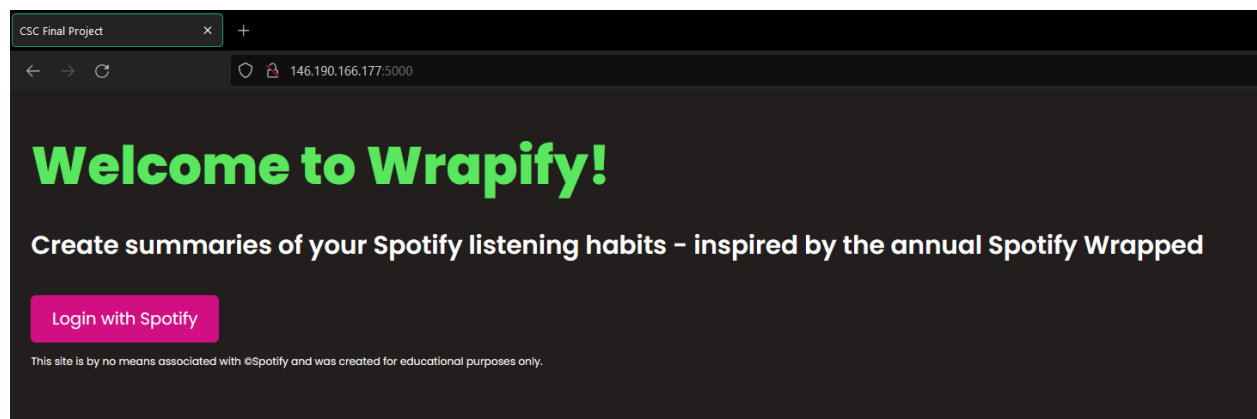
05.03.2024

Final Project Report

Application Use

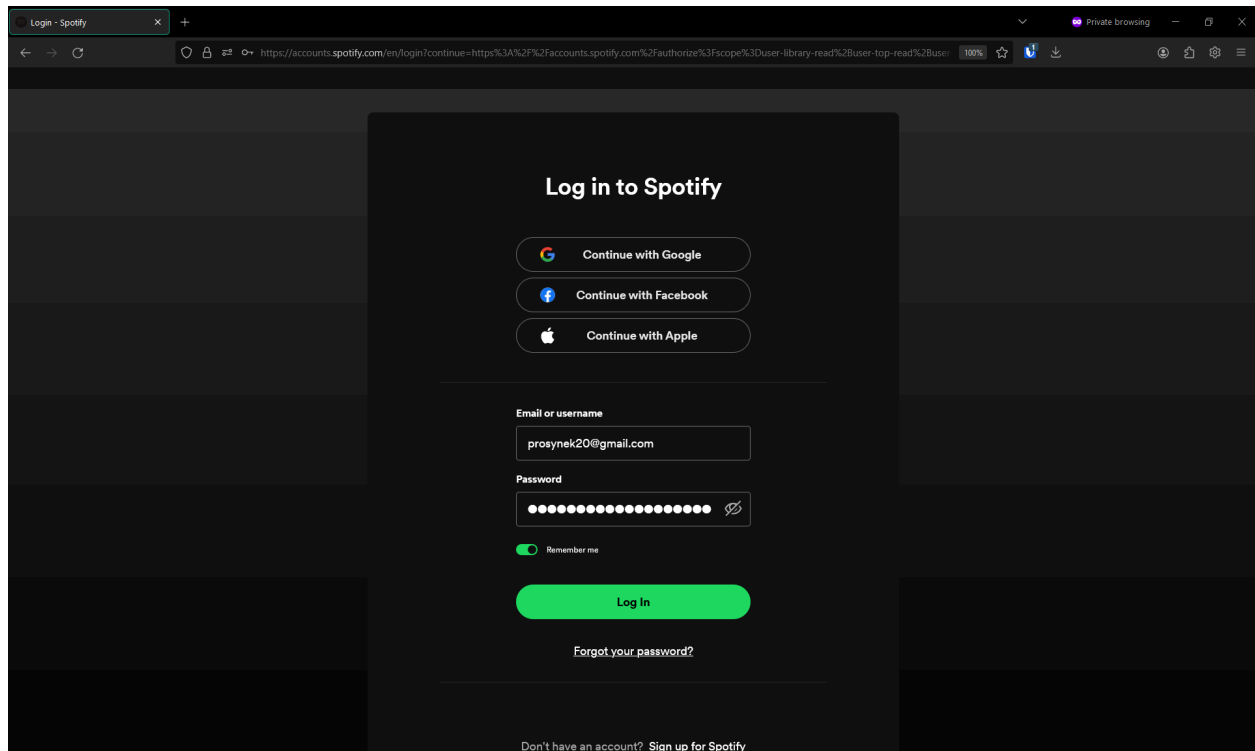
The final application consists of 3 containerized services orchestrated using docker-compose and is deployed on a DigitalOcean Droplet (<http://146.190.166.177:5000/>). The client service can be found in the app_service folder of the project. The app service is responsible for running the application and communicating with the other services and the MongoDB database. The MongoDB database is used to store users' summaries and runs on one of Mongo's Atlas clusters. In addition to the app service, there is an authentication service and Spotify service. The authentication service manages and distributes access tokens using Spotify's OAuth endpoint. The Spotify service is responsible for making requests and retrieving data from Spotify's API. The screenshots below demonstrate how the application works.

Homepage



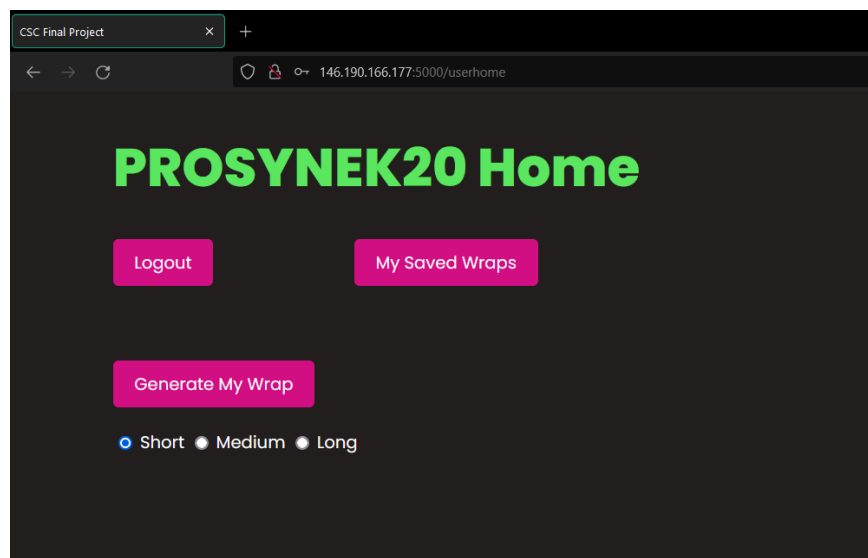
User Login

When a user clicks on 'Login with Spotify' the app service makes a request to the authentication service that will redirect the user to Spotify's account login page.



User Home

After successfully logging into Spotify, the authentication service redirects back to the client application and returns the user to their home page. The title of the page is replaced with their Spotify user ID (e.g. mine is prosynek20).

















Generate Wrap (Summary)






The main functionality of my application is the ability to retrieve and construct a summary or 'wrap' of a Spotify user's listening habits on the app. There are 3 types of wraps a user can generate: short, medium, and long. These values correspond to the time frame of how far back to go when constructing a user's listening summary. The corresponding time frames are as follows:

- short: last ~4 weeks
- medium: last ~6 months
- long: last ~1 year

A user can select a radio button and then click on 'Generate My Wrap' to retrieve their summary which contains information on the user's top 10 tracks, top 10 artists, and top 5 genres from the specified time frame. The app service makes multiple requests to the Spotify service to achieve this. Below is an example of a short-term wrap of my listening habits which is rendered on the same page below the 'Generate My Wrap' button. Every time a user generates a wrap it is saved to the MongoDB database.

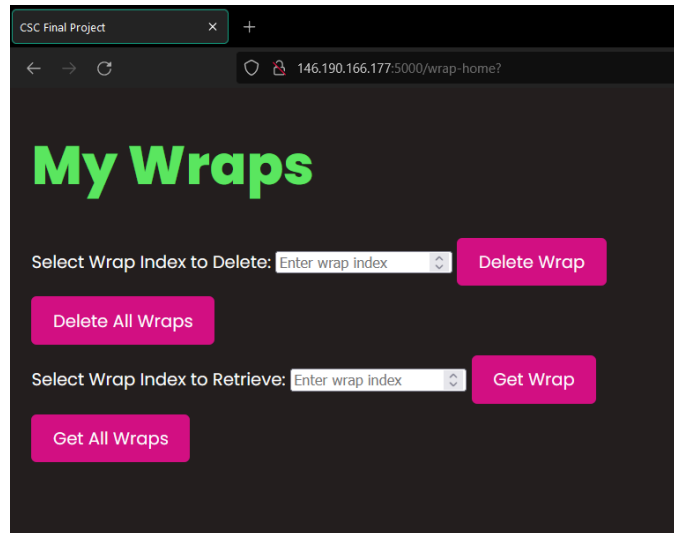
Top Tracks		Duration
	Timezones Schur	03:33
	Bullitt - Bonus Track Elephanz	03:19
	Time > Breathe Reprise > Greensky Bluegrass	05:38
	It's Not Right But It's Okay Mr. Belt & Wezol	02:32
	I Saw You Close Your Eyes Local Natives	03:37
	I'm Fine Apashe, High Klassified, Cherry Lena	03:14
	Sleepy Pietro Machinedrum, Tigran Hamasyan	03:26

	Paper Trail\$ Joey Bada\$\$	03:15
	Seabird Alessi Brothers	03:09
Top Artists		Popularity
	Kendrick Lamar	87
	Montell Fish	72
	Travis Scott	90
	Hozier	84
	Beyoncé	88

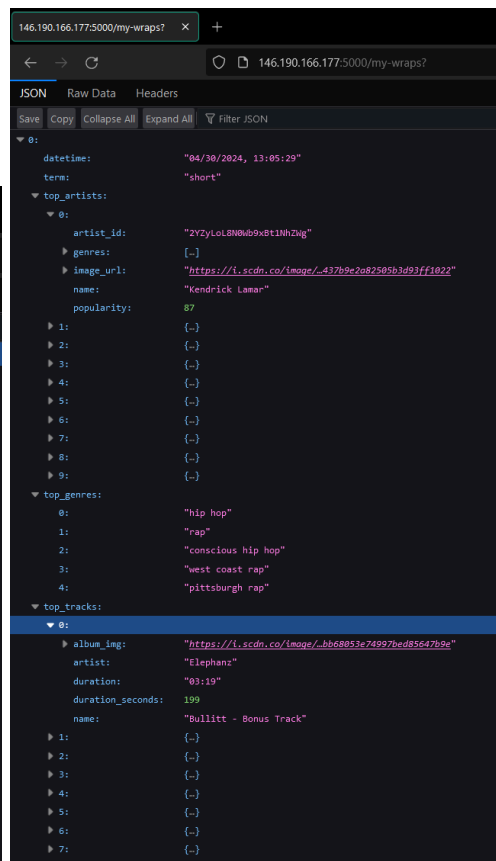
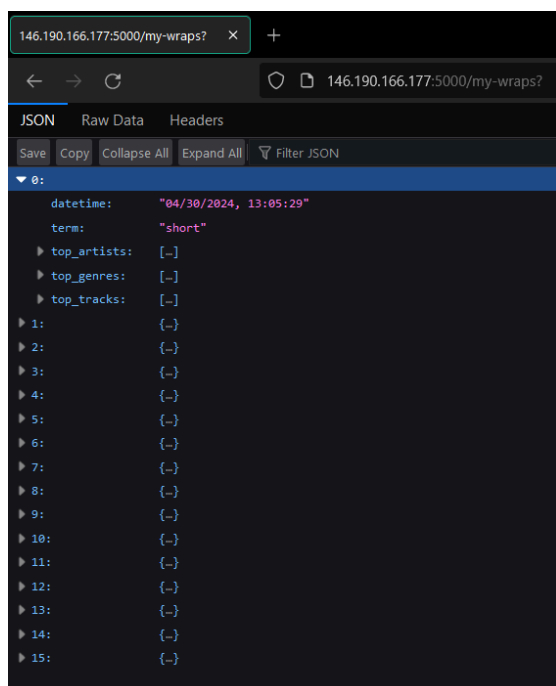
	Nathaniel Rateliff & The Night Sweats	56
	NoMBe	54
	Schur	43
	JAY-Z	80
	Flowervillain	35
Top Genres		
<ul style="list-style-type: none"> • hip hop • rap • conscious hip hop • west coast rap • pittsburgh rap 		

My Saved Wraps

On the user home page, there is a button labeled 'My Saved Wraps' which will navigate the user to the page below. On this page, the user can view and delete all of their past generated wraps that have been stored in the MongoDB database. Each of the endpoints on this page returns JSON as a response.



Get All Wraps



Get Wrap By Index

CSC Final Project

← → ↻ 146.190.166.177:5000/wrap-home?

My Wraps

Select Wrap Index to Delete: Delete Wrap

Delete All Wraps

Select Wrap Index to Retrieve: Get Wrap

Get All Wraps

146.190.166.177:5000/my-wraps

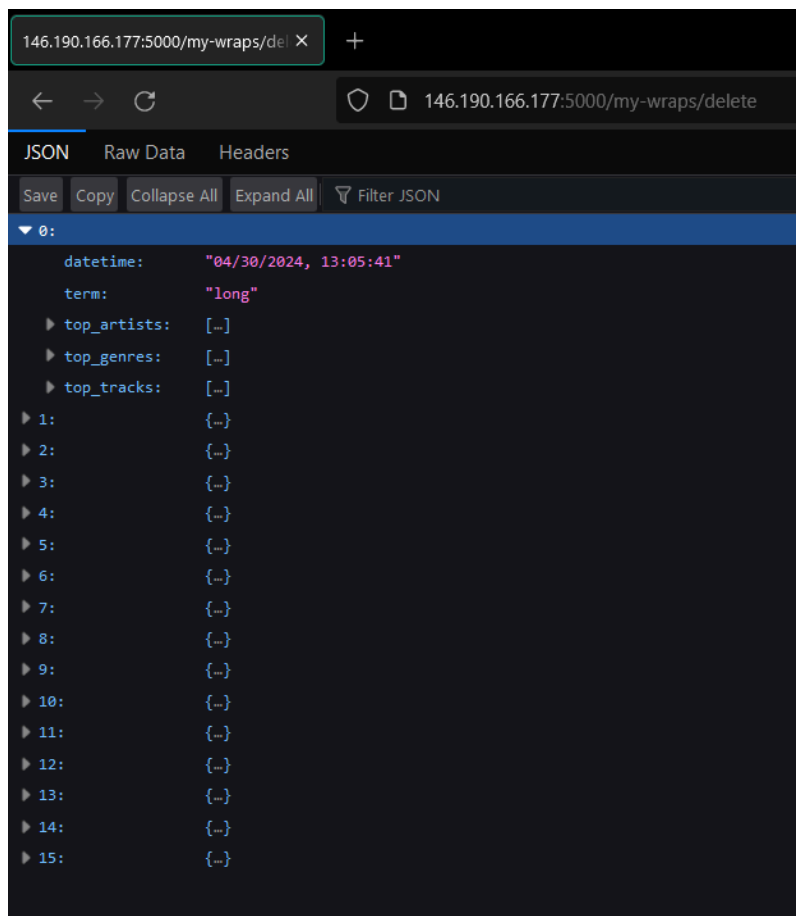
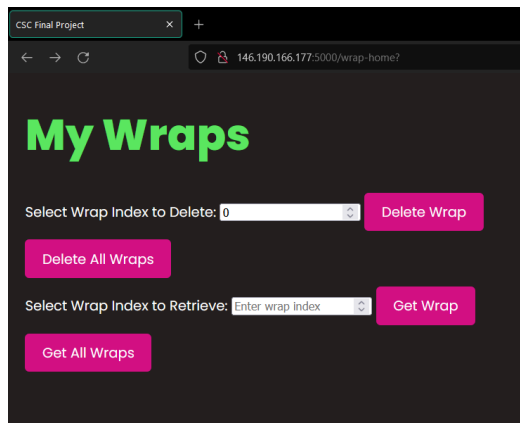
← → ↻ 146.190.166.177:5000/my-wraps

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

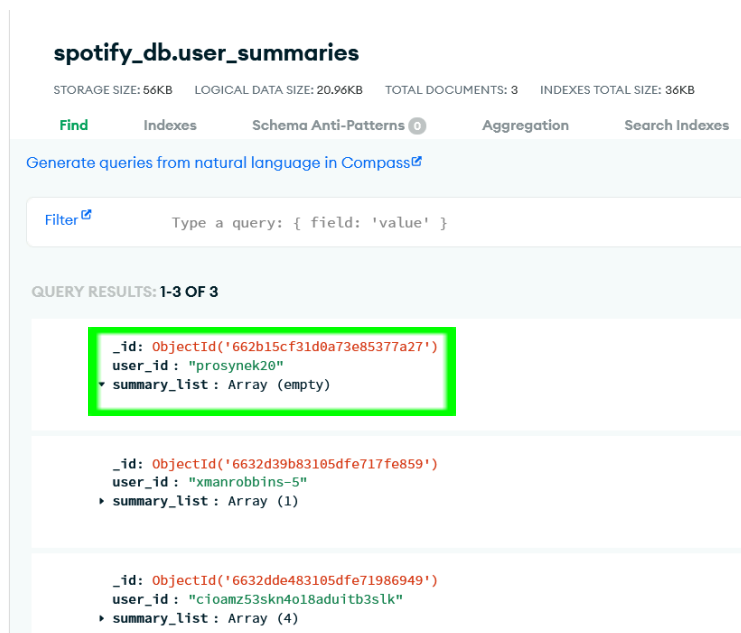
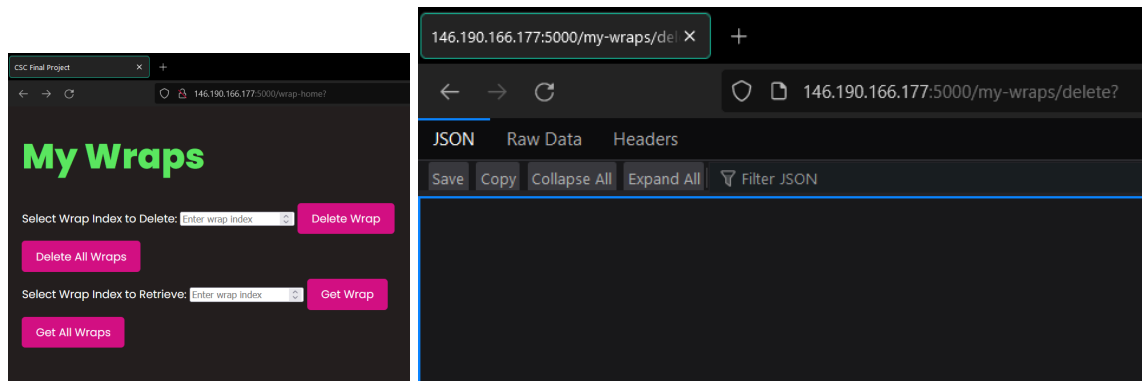
```
datetime: "04/30/2024, 13:05:29"
term: "short"
▼ top_artists:
  ▼ 0:
    artist_id: "2YzyLoL8N0w0b9x0t1NhZWg"
    genres: [...]
    image_url: "https://i.scdn.co/image/_437b9e2a82505b3d93ff1022"
    name: "Kendrick Lamar"
    popularity: 87
    ▶ 1: (...)
    ▶ 2: (...)
    ▶ 3: (...)
    ▶ 4: (...)
    ▶ 5: (...)
    ▶ 6: (...)
    ▶ 7: (...)
    ▶ 8: (...)
    ▶ 9: (...)
  ▼ top_genres:
    0: "hip hop"
    1: "rap"
    2: "conscious hip hop"
    3: "west coast rap"
    4: "pittsburgh rap"
  ▼ top_tracks:
    ▼ 0:
      album_img: "https://i.scdn.co/image/_bb68053e74997bed85647b9e"
      artist: "Elephant"
      duration: "03:19"
      duration_seconds: 199
      name: "Bullitt - Bonus Track"
      ▶ 1: (...)
      ▶ 2: (...)
      ▶ 3: (...)
      ▶ 4: (...)
      ▶ 5: (...)
      ▶ 6: (...)
      ▶ 7: (...)
      ▶ 8: (...)
      ▶ 9: (...)
```

Delete Wrap By Index



Note item 0 is now different indicating it was removed from the list.

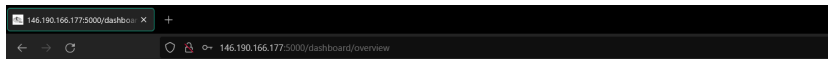
Delete All Wraps



App Monitoring Dashboard

I used a Flask monitoring library called Flask-MonitoringDashboard to record and visualize statistics such as the request count and median request latency for every service endpoint. The dashboard is password-protected so it can only be accessed by authorized users. In addition, I set up this monitoring on each of my services as shown below. The monitoring dashboard has multiple visualizations and metrics available, however, the screenshots below only depict a few of the most important ones.

App Service



Flask-MonitoringDashboard

Automatically monitor the evolving performance of Flask/Python web services

Login

Login

admin

Password

Login

For advanced documentation, see [this site](#)

Flask-MonitoringDashboard

146.190.166.177:5000/dashboard/overview

Private browsing

Flask Monitoring Dashboard

Automatically monitor the evolving performance of Flask/Python web services

Logout

Dashboard

Overview

Hourly API Utilization

Multi Version API Utilization

Daily API Utilization

API Performance

Reporting

Configuration

Dashboard Overview

Number of hits

Median request duration (ms)

Show 10 entries

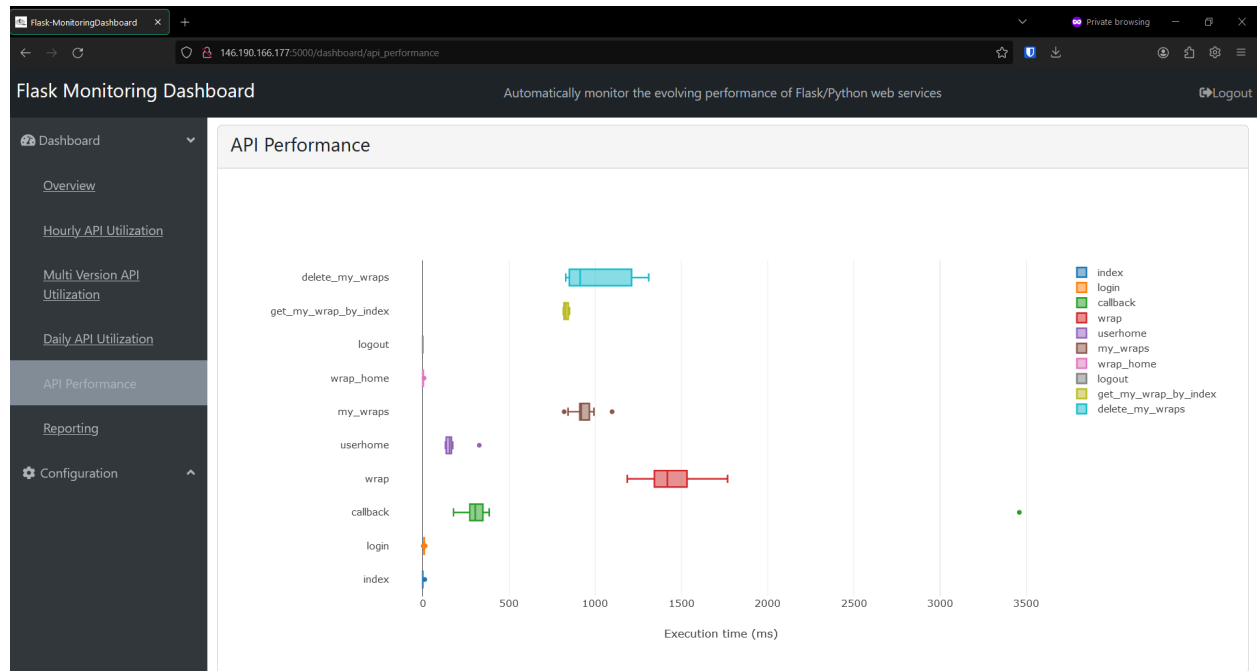
Blueprint:

Search:

Endpoint	Today	Last 7 days	Overall	Last requested	Monitoring-level
index	6	555	555	32 seconds ago	0 1 2 3
login	3	25 ¹	25	31 seconds ago	0 1 2 3
callback	3	25	25	30 seconds ago	0 1 2 3
wrap	4	22	22	5 hours ago	0 1 2 3
userhome	3	16	16	30 seconds ago	0 1 2 3
my_wraps	4	10	10	1 minute ago	0 1 2 3
wrap_home	3	7	7	23 seconds ago	0 1 2 3
get_my_wrap_by_index	4	4	4	14 seconds ago	0 1 2 3
logout	0	4	4	2 days ago	0 1 2 3
delete_my_wraps	2	3	3	5 hours ago	0 1 2 3

Previous

Next



Authentication Service

Flask Monitoring Dashboard

Automatically monitor the evolving performance of Flask/Python web services

Logout

Dashboard

- Overview
- Hourly API Utilization
- Multi Version API Utilization
- Daily API Utilization
- API Performance
- Reporting

Dashboard Overview

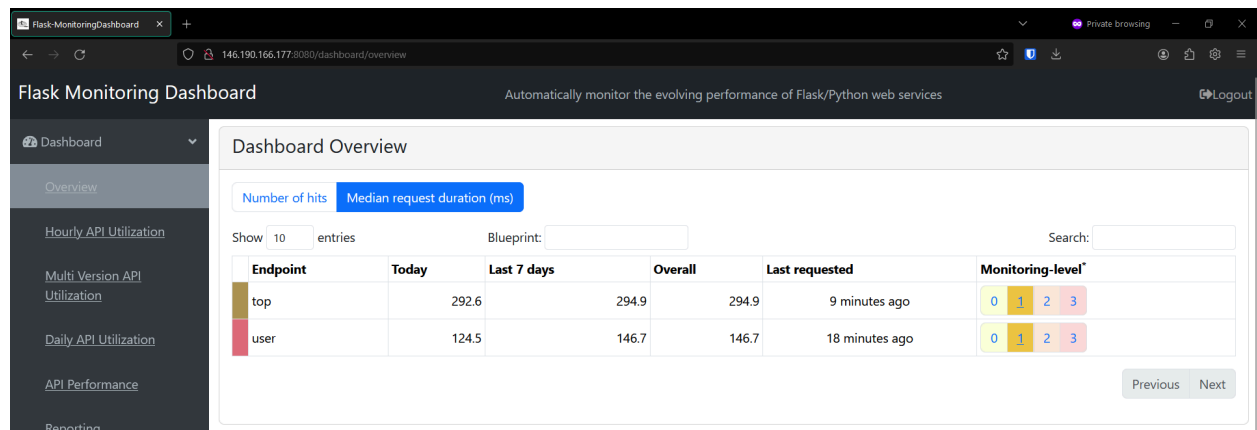
Number of hits | Median request duration (ms)

Show 10 entries | Blueprint: | Search:

Endpoint	Today	Last 7 days	Overall	Last requested	Monitoring-level*
token	1	22	22	17 minutes ago	0 1 2 3
authorize	1	21	21	13 hours ago	0 1 2 3

Previous Next

Spotify Service

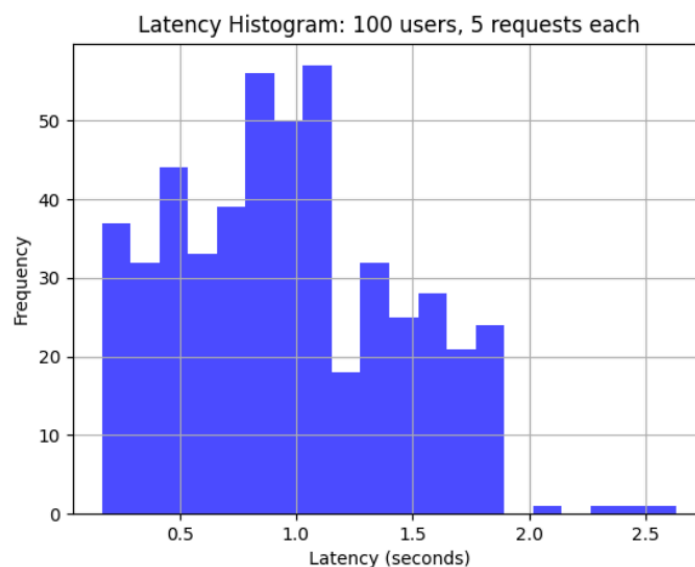


Note to Professor

If you are a Spotify user and would like to try out my application for yourself please send me a Teams message with your Spotify email and I can add you to the whitelist of users. Spotify allows users to create apps in 'development mode' which allows for up to 25 users to use the app, however, I must first add their email to a whitelist before they are allowed access. If you are interested, let me know! :)

Performance Report

The histogram below represents the request latency of the application. The test run simulated 100 users making 5 requests to the home page of my application. The code used to generate this report can be found in the project root in the file named `load_test.py`.



Project Post-Mortem

Overview

Over the course of this project, I was able to successfully design, develop, dockerize, and deploy a simple microservice on a cloud computing platform. My goal for this project was to expand on my Flask and docker skills as well as improve my understanding of cloud computing environments.

Successes

- **Deployment:** I successfully wrote Dockerfiles for each of my services and a docker-compose.yml to orchestrate them. With this, I was able to deploy my dockerized application on a DigitalOcean Droplet running Ubuntu to demonstrate the effective use of containerization and cloud computing.
- **Functionality:** I successfully implemented Spotify's OAuth authentication workflow to enable authentication across my application.
- **Flask:** I improved my knowledge of Flask and I learned how to run Python code from an HTML page.

Challenges

- **Deployment:** I ran into many challenges when determining where and how to deploy my containerized application. After I had my application running with docker-compose I realized I was completely lost on how to actually deploy an application that had been containerized in this way. Trying to deploy my application was a big source of confusion as my research suggested different things and there were numerous cloud platforms I could have used. Initially, I attempted to deploy my app on AWS with ElasticBeanstalk, however after a lot of troubleshooting I opted to go with DigitalOcean and one of their Droplet containers.
- **MongoDB:** Another challenge I faced was with MongoDB. After deploying my application on DigitalOcean, I was unable to access the user database on MongoDB which caused much of my application to be nonfunctional. I spent a lot of time trying to debug this. I initially thought the problem was with the arguments being passed into the MongoClient() object so I made changes to this. Later I learned that there was an access list in MongoDB where I could whitelist specific IPs to allow access to the database. I added the IP of my

DigitalOcean Droplet to the list but failed to revert the changes I made in my code, so I was still experiencing errors. After a lot of frustration, I did a fresh pull of my GitHub repository and everything worked fine indicating that whitelisting the IP was the fix and any changes I made only created more problems for myself. This whole process ended up being a big forehead slap, but it was also a valuable reminder to only change one thing at a time when debugging code.

- **Spotify OAuth:** It was a learning curve to understand and implement OAuth in my application. The documentation from Spotify was very informative, however, I struggled to conceptualize the design of the architecture of my application and how to implement token management.

Lessons Learned

- **Development process:** When beginning to design my microservice application, I struggled to divide my app into microservices. However, I found it was much easier to build a basic, working monolithic application and then later break up the monolith into microservices.
- **OAuth code flows and redirect URIs:** I learned that understanding the concept of OAuth is essential to implementing it correctly. It is especially important to set the correct redirect URI so that after authentication, the user is directed back to your app and not somewhere else. This project helped solidify my understanding of tokens and the process of refreshing expired tokens in the context of OAuth.
- **Spotify development restrictions:** I learned that creating a developer app in Spotify does not simply allow any Spotify user to use your app. Instead, Spotify grants you up to 25 users to use your application. However, they require that you whitelist the email of the users that you want to allow access. I found this out the hard way after I sent my sister the link to my application and she told me it didn't work. I was very afraid my application was broken, however, after some research, I was relieved to learn it was a problem with Spotify and not my implementation.

Future Work

One improvement I would like to make to my app in the future is the ability for a user to save the top tracks from their wrap to a Spotify playlist. I think it would make the application more intriguing and useful. Unfortunately, I ran out of time to make this

addition after I ran into various roadblocks throughout development and deployment. In the future, I would also like to make some changes to the 'My Wraps' page. I think it would improve the user experience if all of the user's past wraps were rendered like they are when they are generated, and then include a delete button for each one where the user could delete a specific wrap without having to know the index of it in the database. Additionally, I would like to learn how to use GitHub actions to set up a CI/CD pipeline for my application. This is something one of my senior design team members set up for our project and I think it is valuable knowledge to have for personal and professional development. I had a lot of fun developing this app and I plan on continuing to work on it in the future.

Project Feedback

This project was challenging yet valuable. One thing that caused me to struggle throughout the project was the lack of structure in the requirements. Although I did appreciate having creative liberty over the topic and tools used for my project, it also caused me to waste a lot of time. It was difficult to come up with a project that was within my ability and would be able to be completed in the given time frame. I found myself coming up with numerous ideas, however with each idea I was unsure if it would fulfill the requirements or could be completed in the time given. I think it would have been helpful to have some example projects listed to help guide students on the scope of the project. One thing I wish we had gone over in class is how to deploy an application that uses docker-compose. The docker-compose lab we did was only run locally so when it came time for me to deploy my application on a cloud service after I had docker-composed it I was extremely lost and my research into the topic was pulling me in different directions. Lastly, I found it strange that failing to complete the proposal would penalize the grade by 10% yet completing it on time would do nothing to count towards the grade. I think it would be fair to incorporate 5-10 points for completing the proposal in the project grade instead of having it simply penalize you for not doing it. However, this is my understanding of the grading according to the project document and I could have potentially misinterpreted this. In general, this project was tough, but an invaluable learning experience nonetheless. Throughout completing this project, I gained confidence with using Docker and docker-compose to containerize applications and I was able to get more comfortable using DigitalOcean. Thank you for the class, have a great summer!

Note: I wasn't sure if you wanted all endpoints of all services documented in the project README so I only included documentation on the app service, there is documentation of the code throughout the project.