Team – Synergy
Soumya Bharathi Vetukuri
Rutuja Patil
Shubham Kothiya
Mann Nada

HW #2 - Mastodon API Integration

Colab Link: https://github.com/BharathiVetukuri/CMPE-272_EnterpriseSoftwarePlatforms/tree/main/Assignment2_Mastodon_API_Integration

Assignment:

The goal of this assignment is to develop a simple service that interacts with the Mastodon API to programmatically create, retrieve, and delete posts. You will also build a basic web UI to demonstrate these functionalities and include appropriate unit tests.

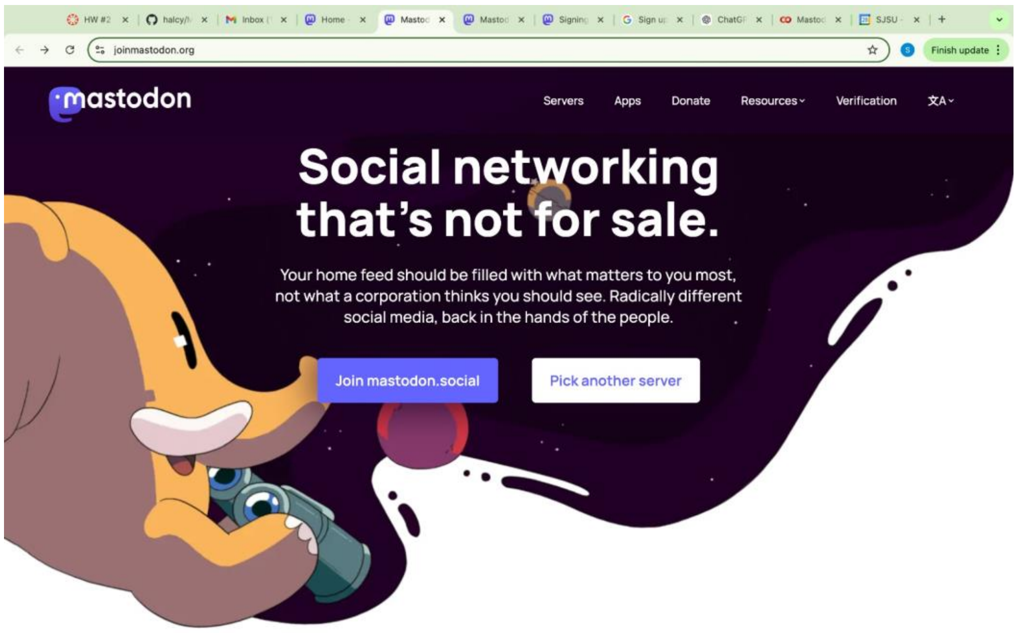
Solution:

What is Mastodon?

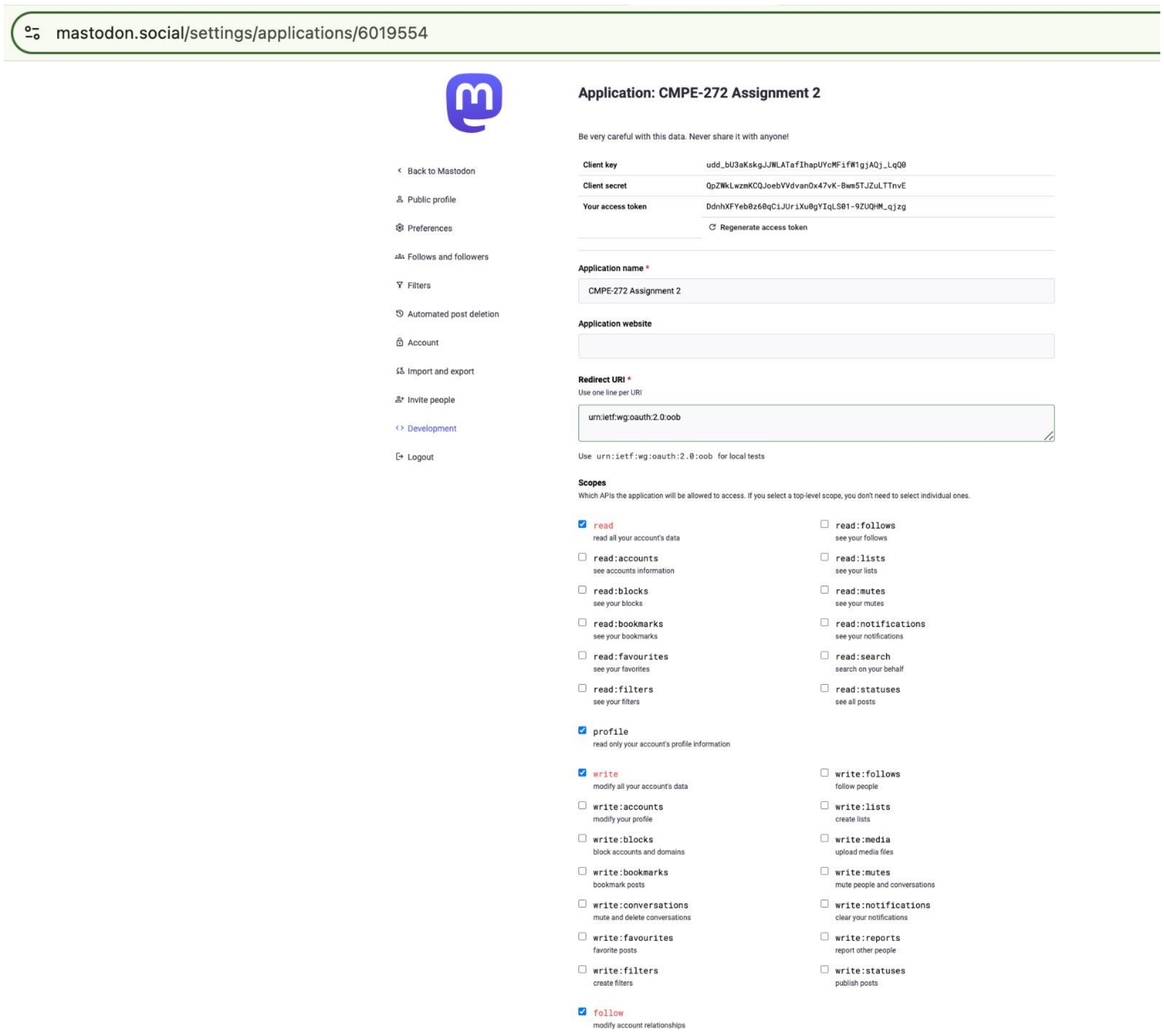
Mastodon is a free, open-source social media platform developed by a non-profit organization and is similar to Twitter. Mastodon is not a single website; to use it we need to make an account with provider called Servers, that lets us connect to other people across Mastodon.

Step 1. Sign Up for a Mastodon Developer Account:

- **Mastodon Home Page:**


joinmastodon.org
mastodon
Servers
Apps
Donate
Resources
Verification
Social networking that's not for sale.
Your home feed should be filled with what matters to you most, not what a corporation thinks you should see. Radically different social media, back in the hands of the people.
Join mastodon.social
Pick another server

- Select a Server: To sign up for a Mastodon account, we chose a server to join: **“Mastodon Social”** then by providing the username and password, can create an account.
- After Login, select the “**Create New Application**” and provide application name and select scopes like read, write follow to give access. Once the application is created, the details page has the API credentials like Access Token and Keys which are essential for the interaction with Mastodon API.



Application: CMPE-272 Assignment 2

Be very careful with this data. Never share it with anyone!

Client key	udd_bU3aKskgJJWLATafIhapUYcMFifW1gjAQj_LqQ0
Client secret	QpZWkLwzmKCQJoebVVdvanOx47vK-Bwm5TJZuLTTnvE
Your access token	DdnhXFYeb0z60qCiJUriXu0gYIqLS01-9ZUQHM_qjzg

Application name *
CMPE-272 Assignment 2

Application website

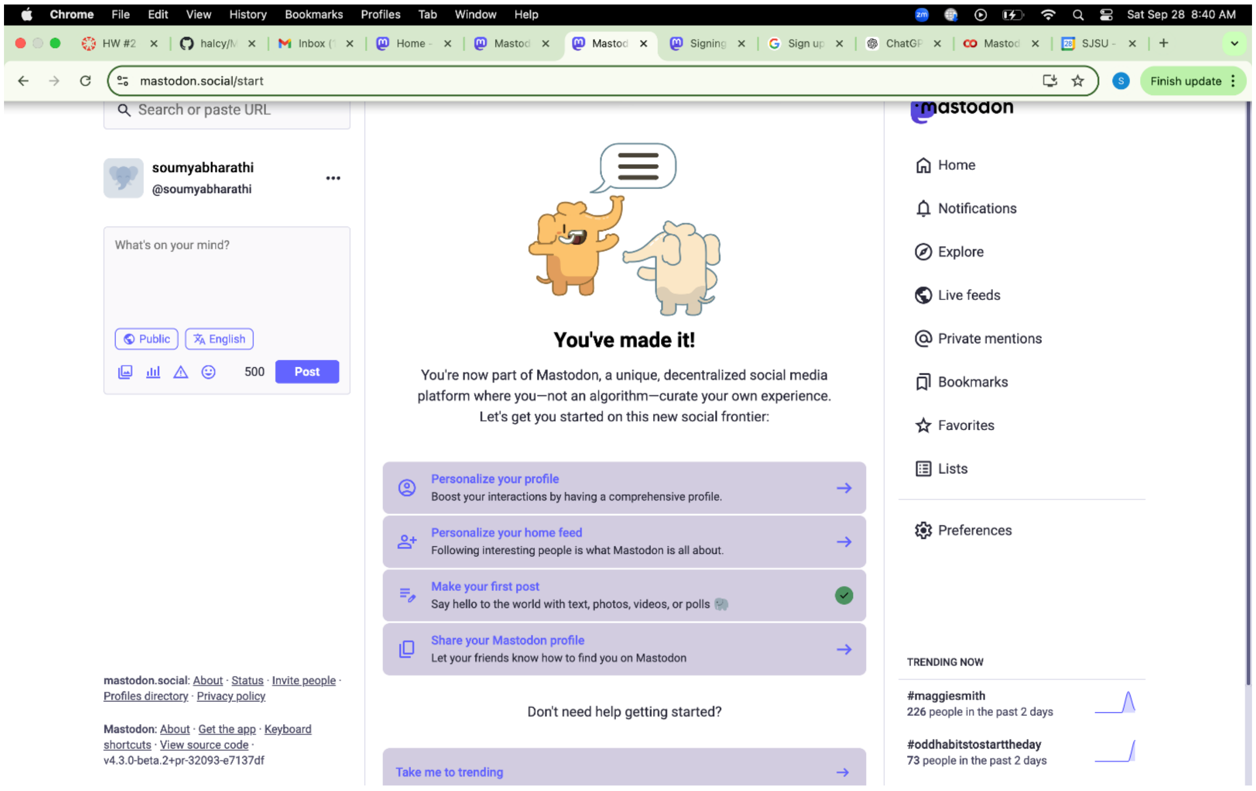
Redirect URI *
Use one line per URI
urn:ietf:wg:oauth:2.0:oob

Use `urn:ietf:wg:oauth:2.0:oob` for local tests

Scopes
Which APIs the application will be allowed to access. If you select a top-level scope, you don't need to select individual ones.

✓ `read` read all your account's data	☐ `read:follows` see your follows
☐ `read:accounts` see accounts information	☐ `read:lists` see your lists
☐ `read:blocks` see your blocks	☐ `read:mutes` see your mutes
☐ `read:bookmarks` see your bookmarks	☐ `read:notifications` see your notifications
☐ `read:favourites` see your favorites	☐ `read:search` search on your behalf
☐ `read:filters` see your filters	☐ `read:statuses` see all posts
✓ `profile` read only your account's profile information	
✓ `write` modify all your account's data	☐ `write:follows` follow people
☐ `write:accounts` modify your profile	☐ `write:lists` create lists
☐ `write:blocks` block accounts and domains	☐ `write:media` upload media files
☐ `write:bookmarks` bookmark posts	☐ `write:mutes` mute people and conversations
☐ `write:conversations` mute and delete conversations	☐ `write:notifications` clear your notifications
☐ `write:favourites` favorite posts	☐ `write:reports` report other people
☐ `write:filters` create filters	☐ `write:statuses` publish posts
✓ `follow` modify account relationships	

- **Mastodon Social Home Page:**



You've made it!

You're now part of Mastodon, a unique, decentralized social media platform where you—not an algorithm—curate your own experience. Let's get you started on this new social frontier:

Personalize your profile
Boost your interactions by having a comprehensive profile.

Personalize your home feed
Following interesting people is what Mastodon is all about.

Make your first post
Say hello to the world with text, photos, videos, or polls

Share your Mastodon profile
Let your friends know how to find you on Mastodon

Don't need help getting started?

Take me to trending

#maggiesmith
226 people in the past 2 days

#oddhabitstostarttheday
73 people in the past 2 days

mastodon.social: About · Status · Invite people · Profiles directory · Privacy policy

Mastodon: About · Get the app · Keyboard shortcuts · View source code · v4.3.0-beta.2+pr-32093-e7137df

Step 2. Develop a Mastodon Service:

Objective: Create a simple service that interacts with the Mastodon API in the following ways:

- **Create a post:** To create a new post (status update) on Mastodon instance.
- **Retrieve a post:** Fetch the post created by querying the API to retrieve the status update.
- **Delete a post:** Delete the post using the Mastodon API.

i. **API Connection:** The Mastodon function is used to setup connection with the service by exposing the access token. The function accepts two input parameters, access token generated from the application created and mastodon social api url since we chose Mastodon Social Service.

```
mastodon = Mastodon(
    access_token = 'DdnhXFYeb0z60qCiJUriXu0gYIqLS01-9ZUQHM_qjzg',
    api_base_url = 'https://mastodon.social'
)
```

ii. **POST (Create Status):** Use the POST method to create a status and display it on the Mastodon Social Home Page of our account. Create_post is the function and it accepts the text to be displayed as the input. Status_post(status_text) is used to post the status on the home page and status['id'] retrieves the unique ID generated for each post. The return statement returns the ID of the post as the response.

```
def create_post(status_text):
    status = mastodon.status_post(status_text)
    st.session_state.post_id = status['id']
    return st.session_state.post_id
```

iii. **GET (Retrieve Status):** The retrieve_post() function retrieves the content of the post generated based on the post_id from the session state. If no post exists, it returns a message indicating that there's nothing to retrieve.

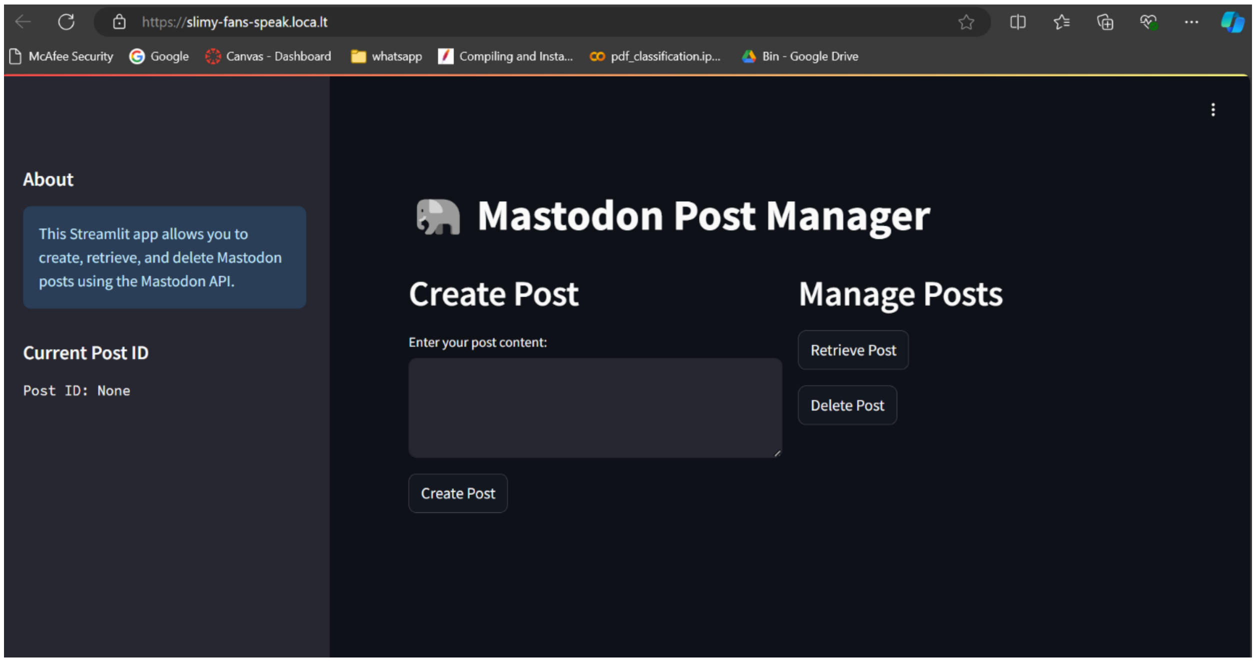
```
def retrieve_post():
    if st.session_state.post_id is None:
        return "No post to retrieve."
    try:
        retrieved_status = mastodon.status(st.session_state.post_id)
        return f"Retrieved post: {retrieved_status['content']}"
    except Exception as e:
        return f"Error retrieving post: {str(e)}"
```

iv. **DELETE (Delete Status):** We also provide the ability to delete a post. The delete_post function removes the post using the post_id and then resets the session state's post_id to None.

```
    def delete_post():
        if st.session_state.post_id is None:
            return "No post to delete."
        mastodon.status_delete(st.session_state.post_id)
        result = f"Deleted Post with ID {st.session_state.post_id}"
        st.session_state.post_id = None
        return result
```

Step 3. Build Web UI:

We built a simple Web UI to demonstrate and visualize the three API functionalities performed above. The ‘Create Post’, ‘Retrieve Post’ and ‘Delete Post’ buttons perform the actions. A ‘Create Post’ text box is available to input the content of the status. Find the screenshot of the UI beneath:



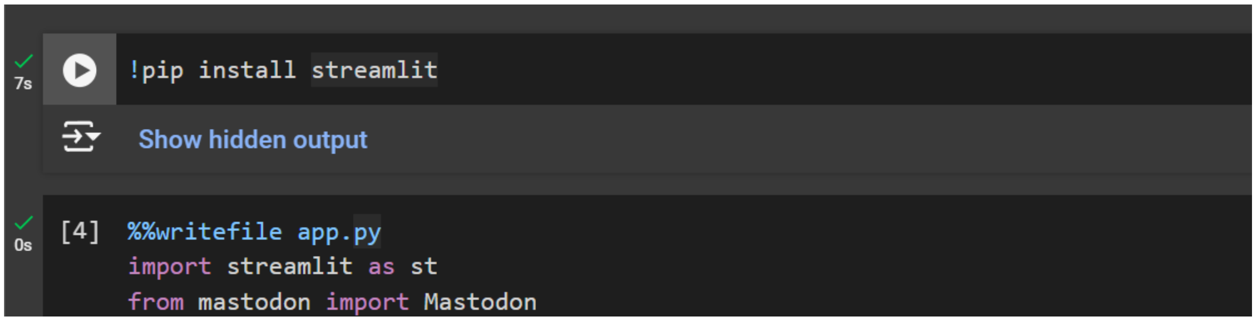
This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.

Post ID: None

Mastodon Post Manager

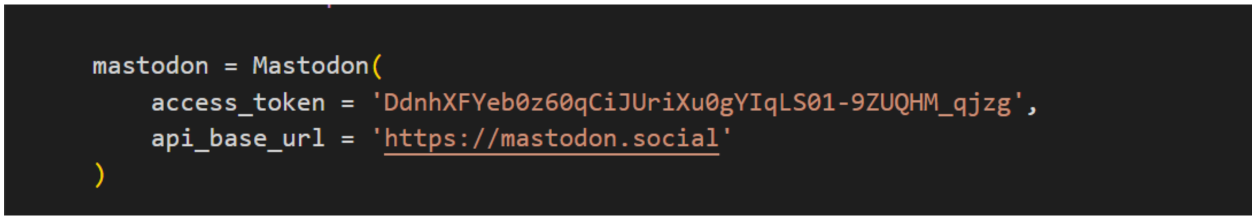
Enter your post content:

i. **Importing Required Libraries:**
We start by importing the necessary libraries. Streamlit is used to build a web interface, and Mastodon enables communication with the Mastodon API.

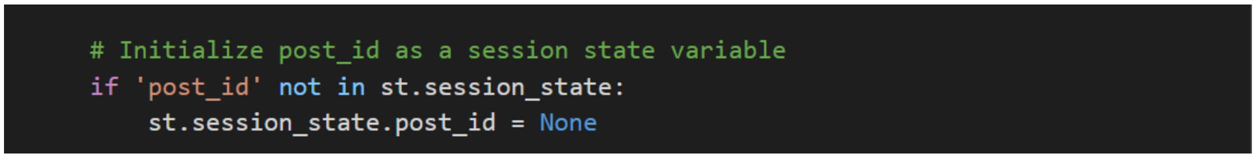

```
!pip install streamlit
```

```
%%writefile app.py
import streamlit as st
from mastodon import Mastodon
```

ii. **Setting Up Mastodon API Access:**
Next, we initialize the Mastodon client using an access token and base API URL for Mastodon. This will allow us to interact with our Mastodon account


```
mastodon = Mastodon(
    access_token = 'DdnhXFYeb0z60qCiJUriXu0gYIqLS01-9ZUQHM_qjzg',
    api_base_url = 'https://mastodon.social'
)
```

iii. **Managing the Post ID with Streamlit Session State:**
The post_id, which refers to the Mastodon post we create, is stored in the session state. This allows us to track the post ID across multiple user interactions within the app.


```
# Initialize post_id as a session state variable
if 'post_id' not in st.session_state:
    st.session_state.post_id = None
```

iv. **Creating a Post:**
This function handles posting content to Mastodon. It uses the status_post method to create a post, stores the resulting post_id in the session state, and returns the post_id.

```python
def create_post(status_text):
    status = mastodon.status_post(status_text)
    st.session_state.post_id = status['id']
    return st.session_state.post_id
```

v. **Retrieving a Post:**
The retrieve_post function retrieves the content of a post using the post_id from the session state. If no post exists, it returns a message indicating that there's nothing to retrieve.

```python
def retrieve_post():
    if st.session_state.post_id is None:
        return "No post to retrieve."
    try:
        retrieved_status = mastodon.status(st.session_state.post_id)
        return f"Retrieved post: {retrieved_status['content']}"
    except Exception as e:
        return f"Error retrieving post: {str(e)}"
```

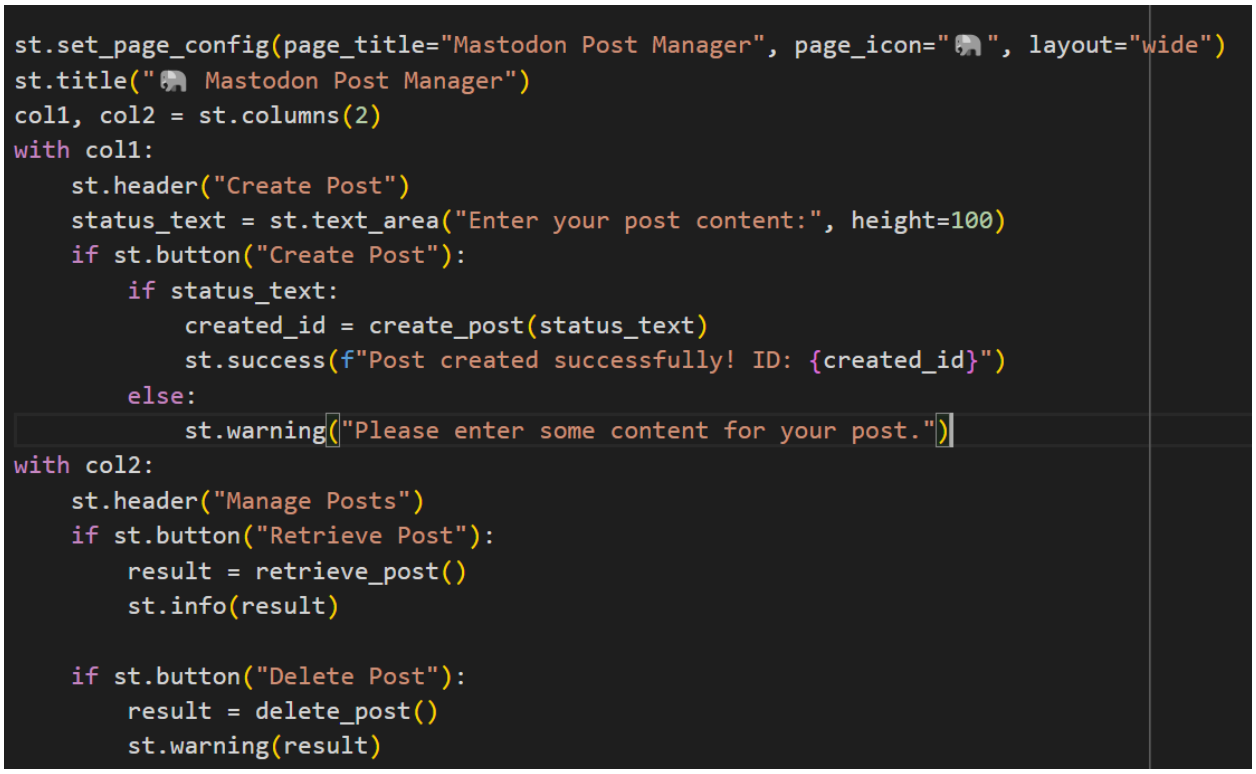
vi. **Deleting a Post:**
We also provide the ability to delete a post. The delete_post function removes the post using the post_id and then resets the session state’s post_id to None.

```python
def delete_post():
    if st.session_state.post_id is None:
        return "No post to delete."
    mastodon.status_delete(st.session_state.post_id)
    result = f"Deleted Post with ID {st.session_state.post_id}"
    st.session_state.post_id = None
    return result
```

vii. **Streamlit Web UI:**
The UI is divided into two columns. The first column is for creating a post, and the second column manages the retrieval and deletion of posts.

I. **Create Post Section:**
A text area allows the user to input post content, and when the "Create Post" button is clicked, the post is created and its ID is displayed.

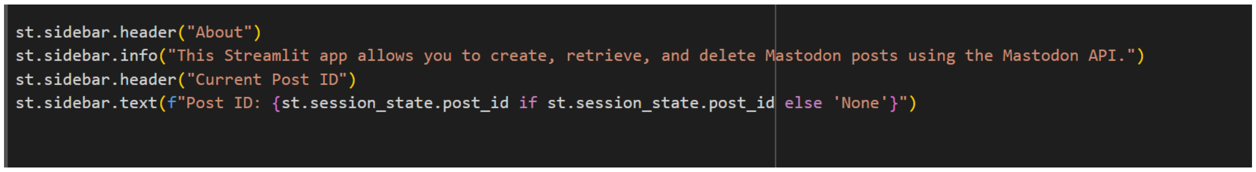
II. **Manage Posts Section:**
The second column contains buttons to retrieve or delete the post. The retrieved post content is shown, or the post is deleted when the respective buttons are pressed.



```python
st.set_page_config(page_title="Mastodon Post Manager", page_icon="🐘", layout="wide")
st.title("🐘 Mastodon Post Manager")
col1, col2 = st.columns(2)
with col1:
    st.header("Create Post")
    status_text = st.text_area("Enter your post content:", height=100)
    if st.button("Create Post"):
        if status_text:
            created_id = create_post(status_text)
            st.success(f"Post created successfully! ID: {created_id}")
        else:
            st.warning("Please enter some content for your post.")
with col2:
    st.header("Manage Posts")
    if st.button("Retrieve Post"):
        result = retrieve_post()
        st.info(result)

    if st.button("Delete Post"):
        result = delete_post()
        st.warning(result)
```

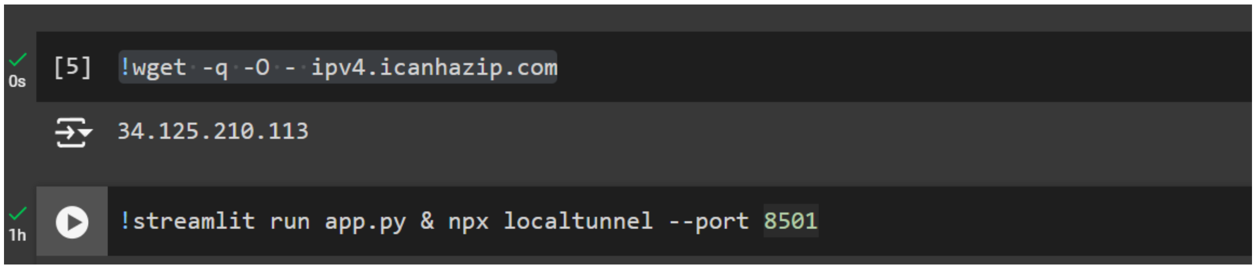
viii. **Sidebar Information**
The sidebar provides helpful information about the app’s functionality and shows the current post_id for easy reference.



```python
st.sidebar.header("About")
st.sidebar.info("This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.")
st.sidebar.header("Current Post ID")
st.sidebar.text(f"Post ID: {st.session_state.post_id if st.session_state.post_id else 'None'}")
```

ix. **Running the App**:

The app is deployed locally using Streamlit, and a tunnel is opened with npx localtunnel to make it accessible via a public URL.



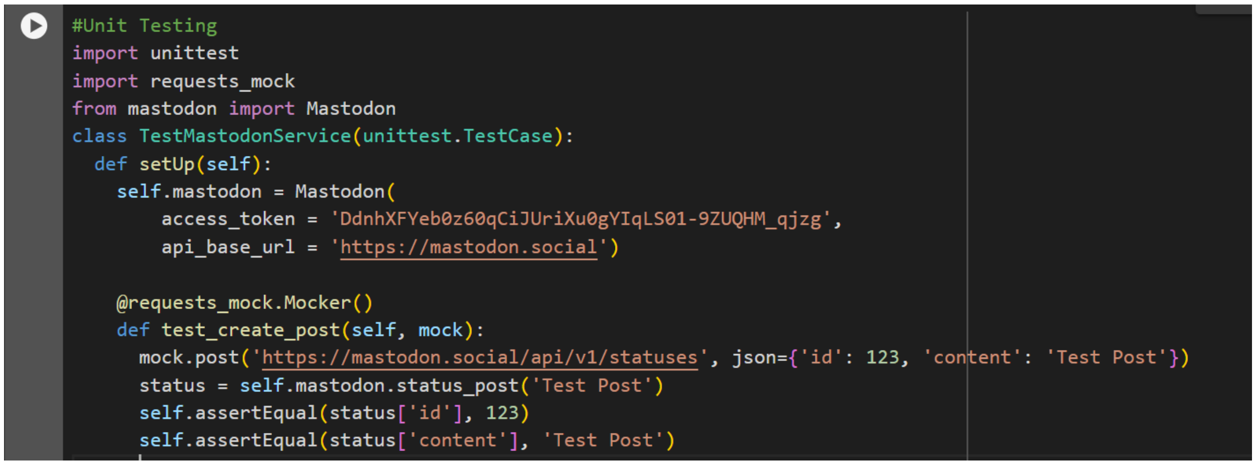
```
[5] !wget -q -O - ipv4.icanhazip.com

34.125.210.113

!streamlit run app.py & npx localtunnel --port 8501
```

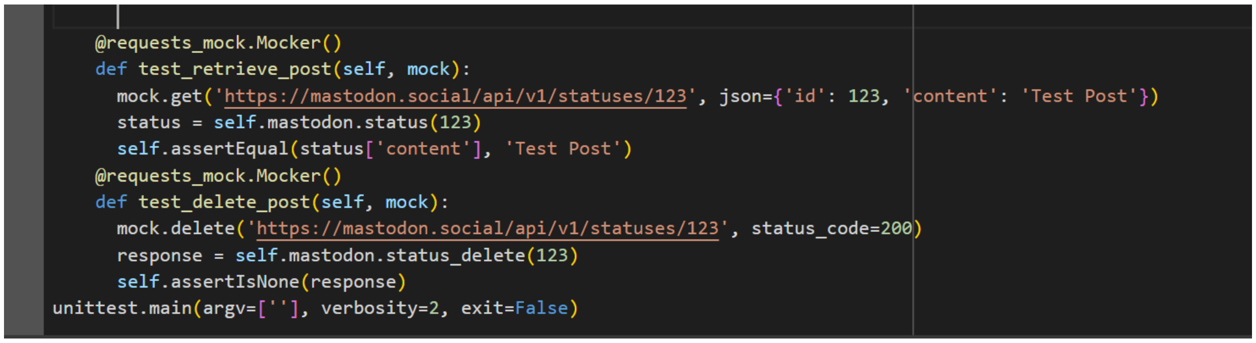
Step 4: Unit Tests:

This code defines unit tests for a Mastodon service using ‘unittest’ and ‘requests_mock’. It mocks the API interactions for creating, retrieving, and deleting posts. The tests check if creating a post returns the correct ID and content, retrieving a post matches the expected content, and deleting a post completes successfully. These tests ensure the Mastodon API functions behave as expected without actual API calls.

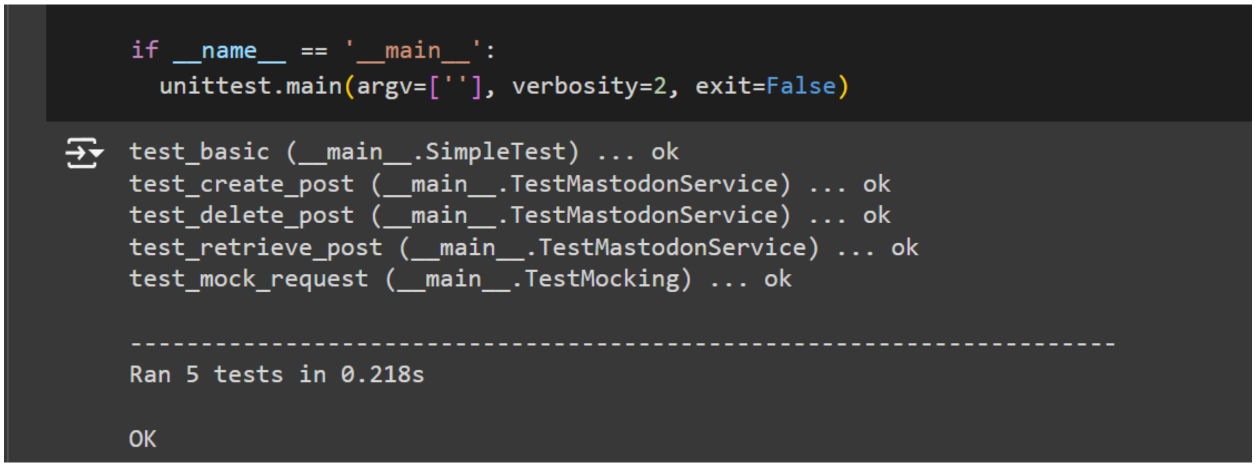


```
#Unit Testing
import unittest
import requests_mock
from mastodon import Mastodon
class TestMastodonService(unittest.TestCase):
  def setUp(self):
    self.mastodon = Mastodon(
        access_token = 'DdnhXFYeb0z60qCiJUriXu0gYIqLS01-9ZUQHM_qjzg',
        api_base_url = 'https://mastodon.social')

    @requests_mock.Mocker()
    def test_create_post(self, mock):
      mock.post('https://mastodon.social/api/v1/statuses', json={'id': 123, 'content': 'Test Post'})
      status = self.mastodon.status_post('Test Post')
      self.assertEqual(status['id'], 123)
      self.assertEqual(status['content'], 'Test Post')
```



```
    @requests_mock.Mocker()
    def test_retrieve_post(self, mock):
      mock.get('https://mastodon.social/api/v1/statuses/123', json={'id': 123, 'content': 'Test Post'})
      status = self.mastodon.status(123)
      self.assertEqual(status['content'], 'Test Post')
    @requests_mock.Mocker()
    def test_delete_post(self, mock):
      mock.delete('https://mastodon.social/api/v1/statuses/123', status_code=200)
      response = self.mastodon.status_delete(123)
      self.assertIsNone(response)
unittest.main(argv=[''], verbosity=2, exit=False)
```



```
if __name__ == '__main__':
  unittest.main(argv=[''], verbosity=2, exit=False)
```

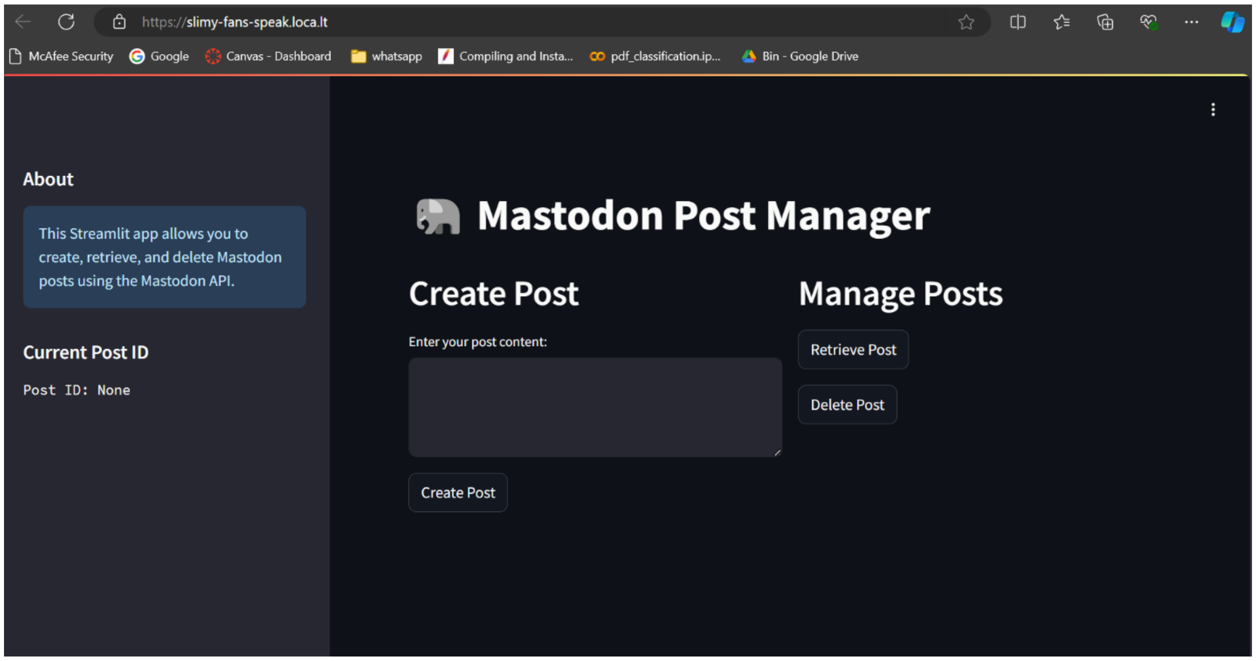
```
test_basic (__main__.SimpleTest) ... ok
test_create_post (__main__.TestMastodonService) ... ok
test_delete_post (__main__.TestMastodonService) ... ok
test_retrieve_post (__main__.TestMastodonService) ... ok
test_mock_request (__main__.TestMocking) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.218s

OK
```

UI Screenshots

The following screenshots demonstrate the step-by-step process of creating a post, Retrieval and Deleting it from the Web UI we created:



https://slimy-fans-speak.loca.lt

About

This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.

Current Post ID

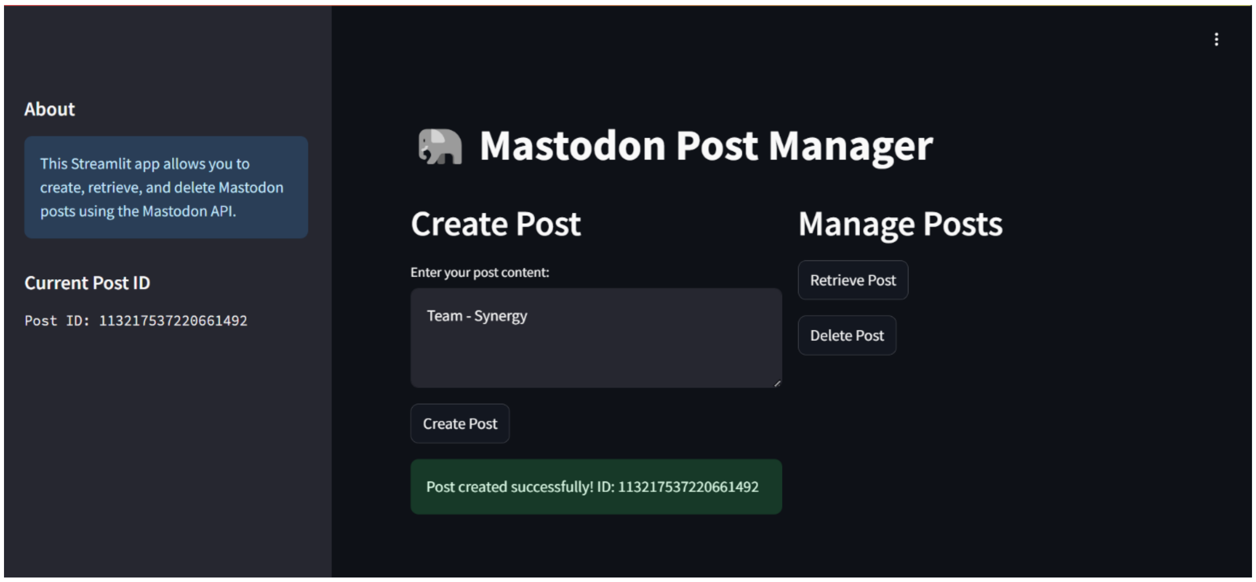
Post ID: None

Mastodon Post Manager

Create Post

Enter your post content:

- **Creating a Status:**



About

This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.

Current Post ID

```
Post ID: 113217537220661492
```

Mastodon Post Manager

Create Post

Enter your post content:

Team - Synergy

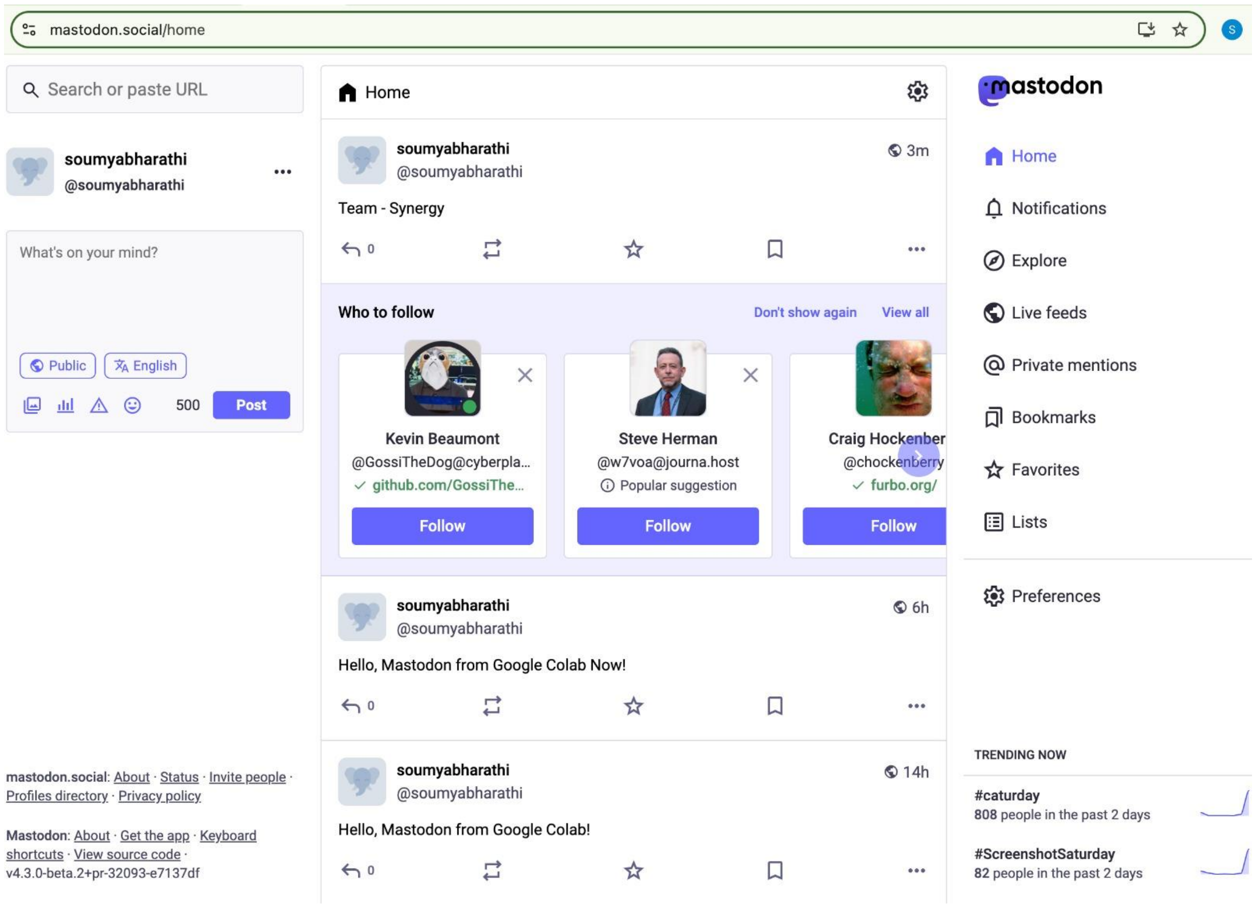
Create Post

Post created successfully! ID: 113217537220661492

Manage Posts

Retrieve Post

Delete Post



mastodon.social/home

soumyabharathi @soumyabharathi · 3m

Team - Synergy

Who to follow

Kevin Beaumont @GossiTheDog@cyberpla… github.com/GossiThe…

Steve Herman @w7voa@journa.host Popular suggestion

Craig Hockenber @chockenberry furbo.org/

soumyabharathi @soumyabharathi · 6h

Hello, Mastodon from Google Colab Now!

soumyabharathi @soumyabharathi · 14h

Hello, Mastodon from Google Colab!

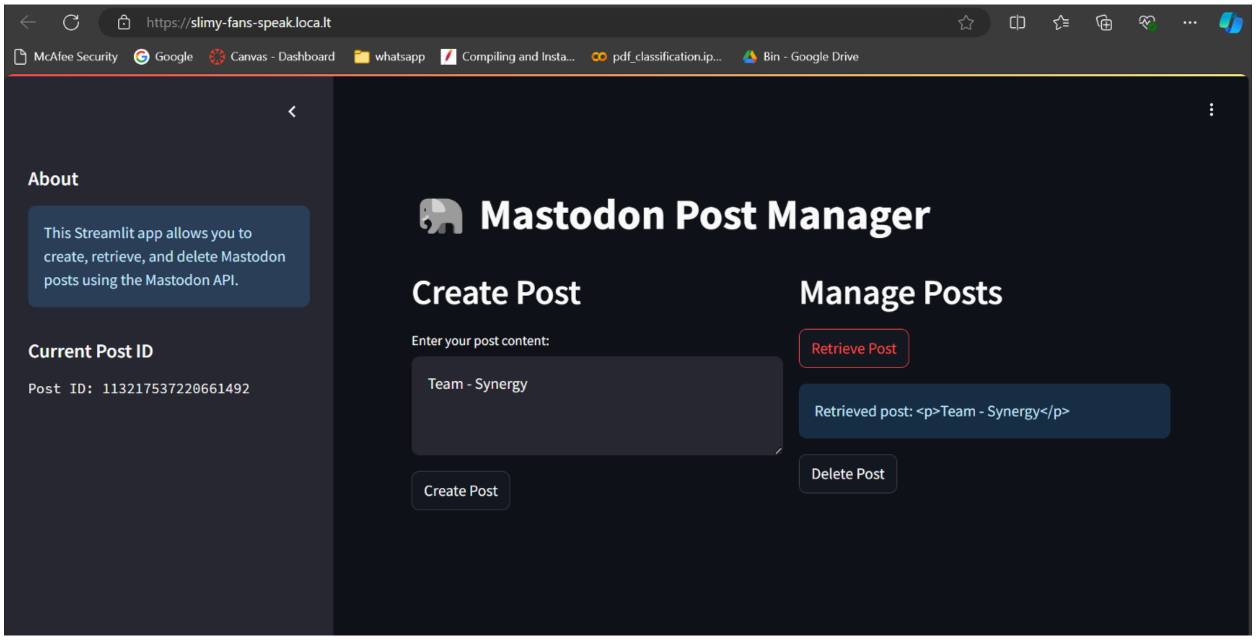
TRENDING NOW

#caturday 808 people in the past 2 days

#ScreenshotSaturday 82 people in the past 2 days

Search or paste URL
What's on your mind?
Public
English
500
Post
Home
Don't show again
View all
Follow
Notifications
Explore
Live feeds
Private mentions
Bookmarks
Favorites
Lists
Preferences
mastodon.social: About · Status · Invite people · Profiles directory · Privacy policy
Mastodon: About · Get the app · Keyboard shortcuts · View source code · v4.3.0-beta.2+pr-32093-e7137df

- **Retrieving a Status:**



About

This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.

Current Post ID

Post ID: 113217537220661492

Mastodon Post Manager

Create Post

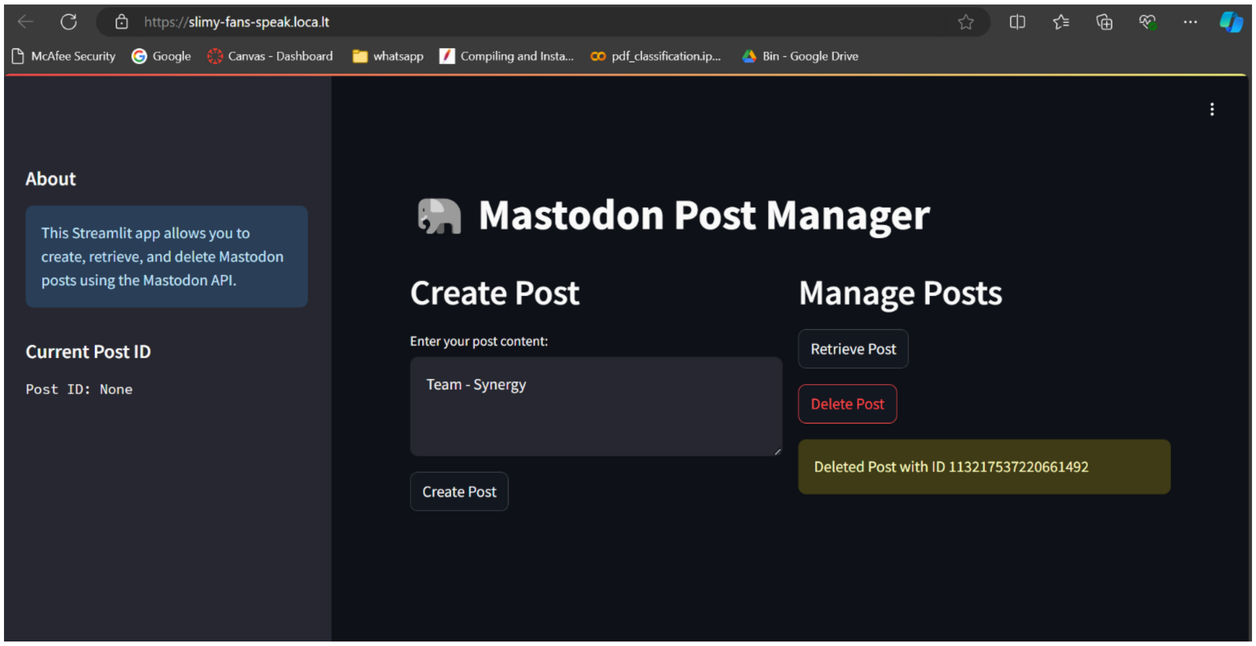
Enter your post content:

Team - Synergy

Manage Posts

Retrieved post: <p>Team - Synergy</p>

- **Deleting a Status:**



About

This Streamlit app allows you to create, retrieve, and delete Mastodon posts using the Mastodon API.

Current Post ID

Post ID: None

Mastodon Post Manager

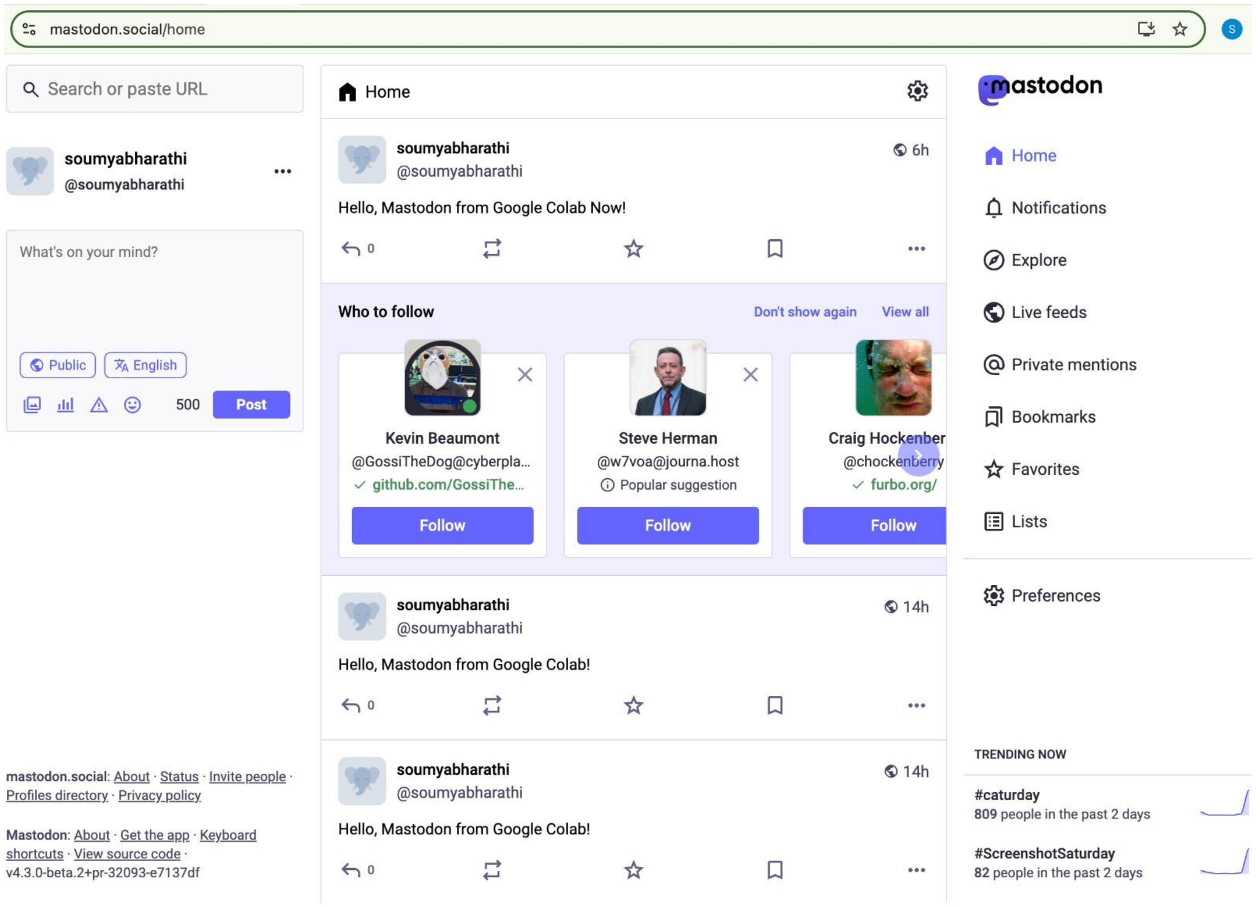
Create Post

Enter your post content:

Team - Synergy

Manage Posts

Deleted Post with ID 113217537220661492



mastodon.social/home

Search or paste URL

soumyabharathi
@soumyabharathi

What's on your mind?

Public English 500 Post

Home

soumyabharathi @soumyabharathi 6h

Hello, Mastodon from Google Colab Now!

0

Who to follow — Don't show again — View all

Kevin Beaumont	Steve Herman	Craig Hockenber
@GossiTheDog@cyberpla...	@w7voa@journa.host	@chockenberry
✓ github.com/GossiThe...	Popular suggestion	✓ furbo.org/
Follow	Follow	Follow

soumyabharathi @soumyabharathi 14h

Hello, Mastodon from Google Colab!

0

soumyabharathi @soumyabharathi 14h

Hello, Mastodon from Google Colab!

0

mastodon

Home
Notifications
Explore
Live feeds
Private mentions
Bookmarks
Favorites
Lists
Preferences

TRENDING NOW

#caturday
809 people in the past 2 days

#ScreenshotSaturday
82 people in the past 2 days

mastodon.social: About · Status · Invite people · Profiles directory · Privacy policy

Mastodon: About · Get the app · Keyboard shortcuts · View source code · v4.3.0-beta.2+pr-32093-e7137df