

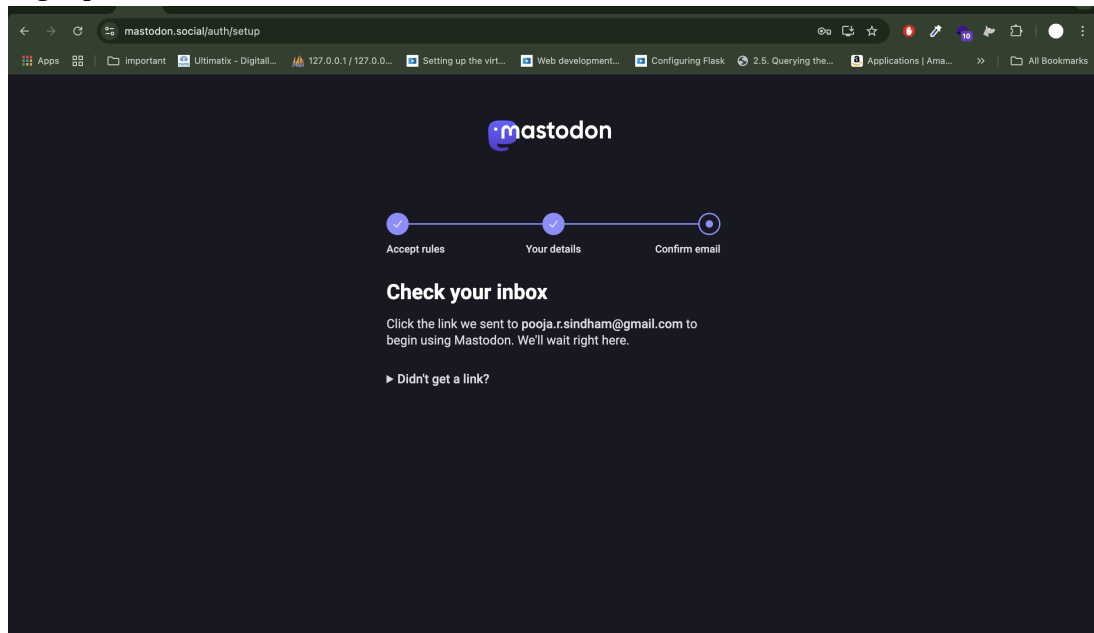
# Homework #2 – Twitter Service

*Aarsh Sheth || Pooja Sindham || Shivani Jariwala || Aishwariya Indi*

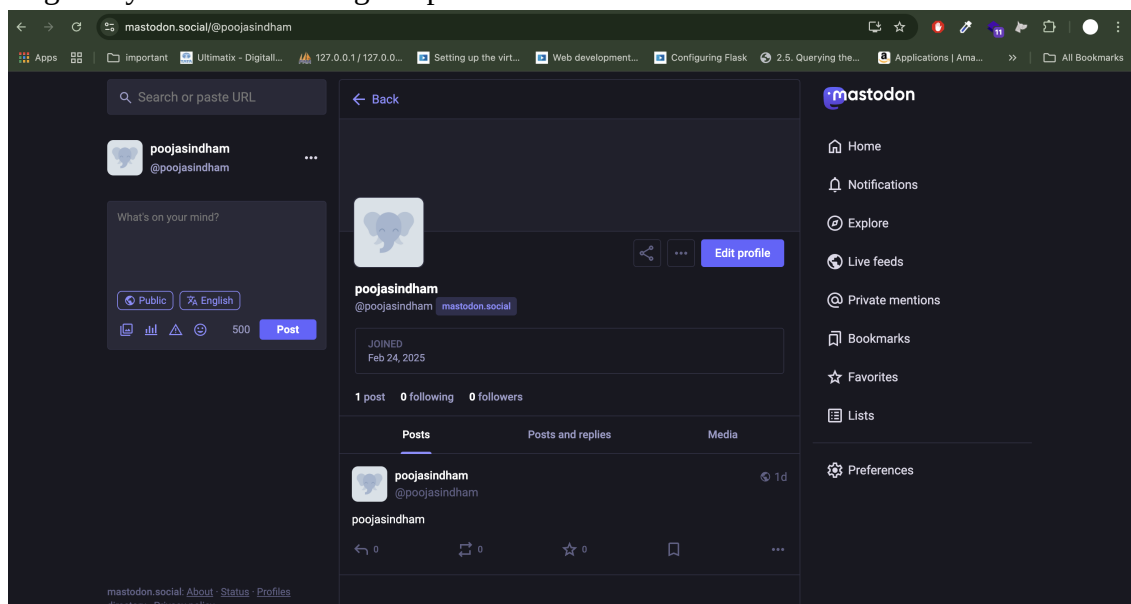
[GitHub Link](#)

## Get Access token for Mostodon Social

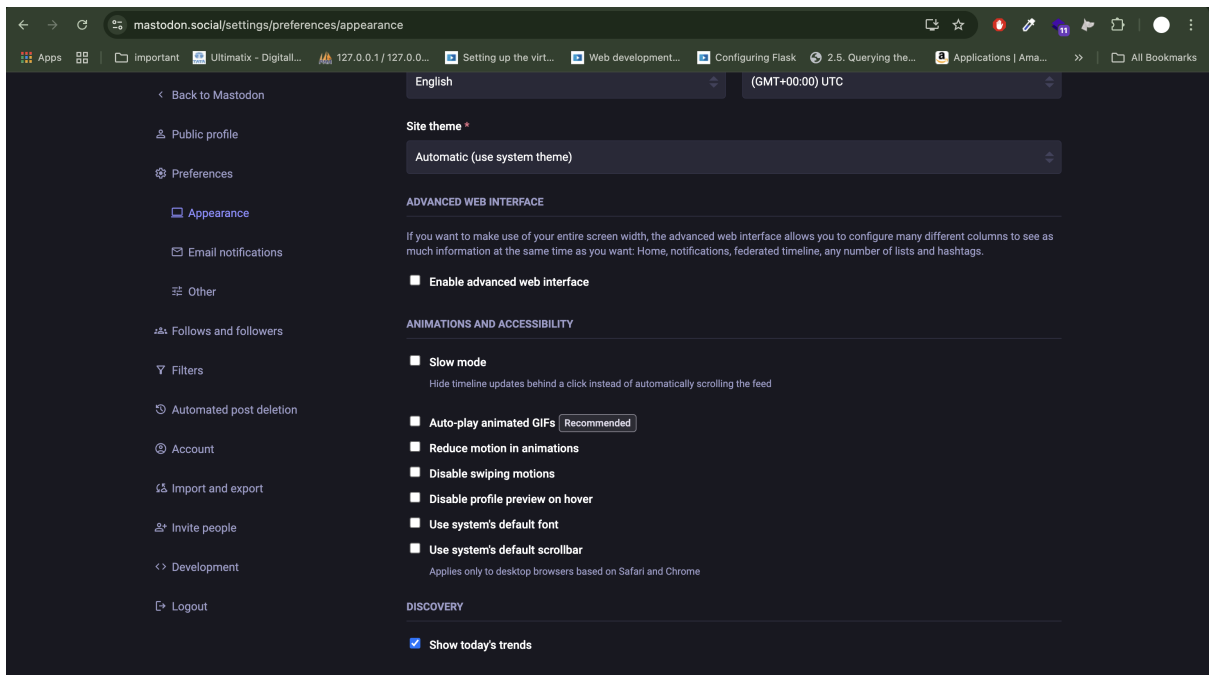
1. Signup for mastodon social.



2. Login to your account and go to preferences.



3. Navigate to Preferences > Development.

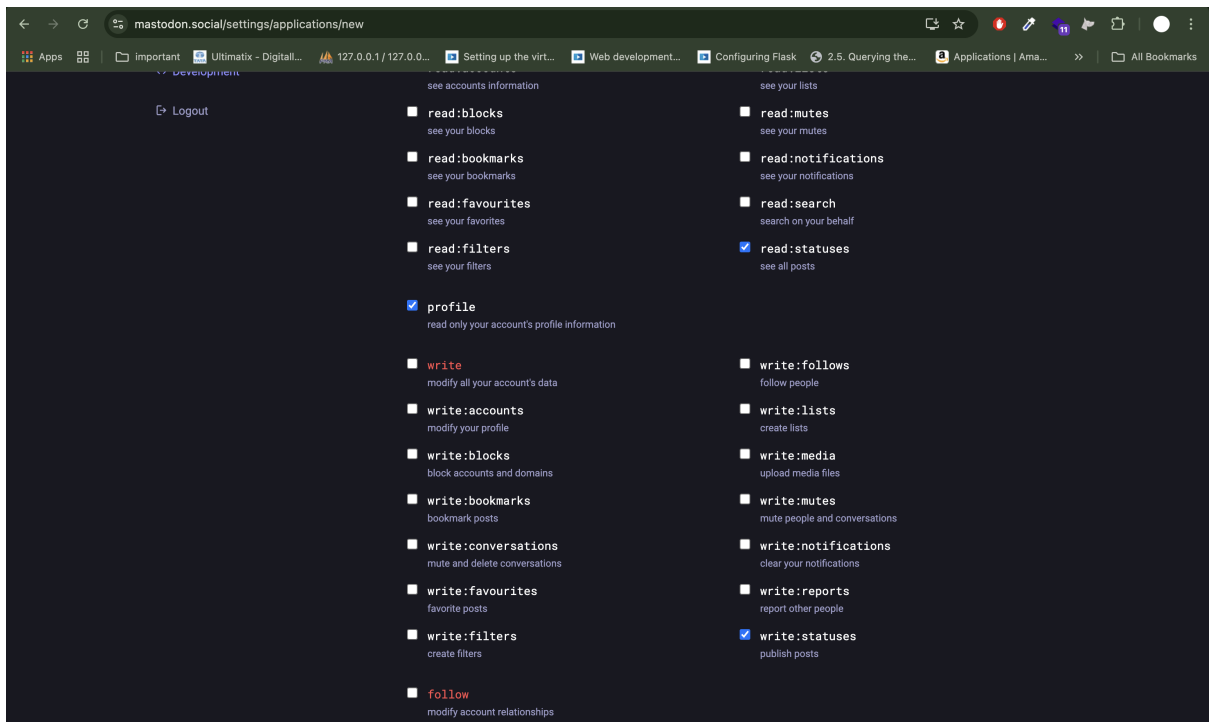


#### 4. Create new application.

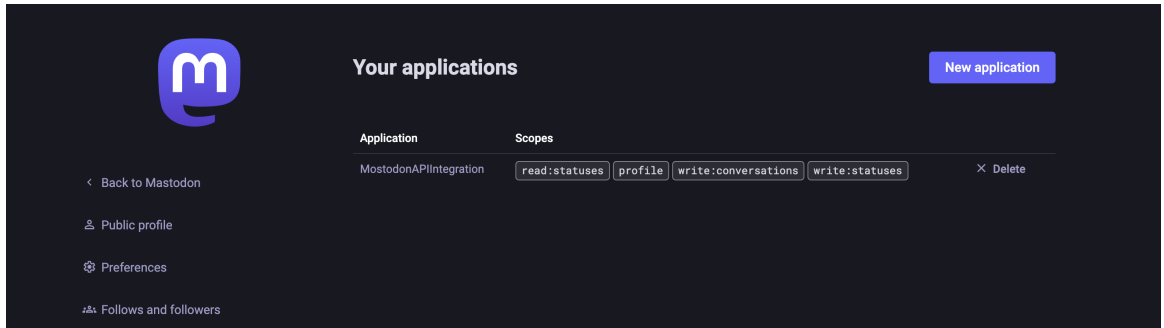
The screenshot shows the 'New application' form in Mastodon. The left sidebar is identical to the previous screenshot. The main content area has the title 'New application' and three input fields: 'Application name' with the value 'MostodonAPIIntegration', 'Application website' which is empty, and 'Redirect URI' with the value 'urn:ietf:wg:oauth:2.0:oob'. A note at the bottom states: 'Use urn:ietf:wg:oauth:2.0:oob for local tests'.

Select the following options

- *read:statuses* (Retrieve posts)
- *write:statuses* (Create and **delete** posts)

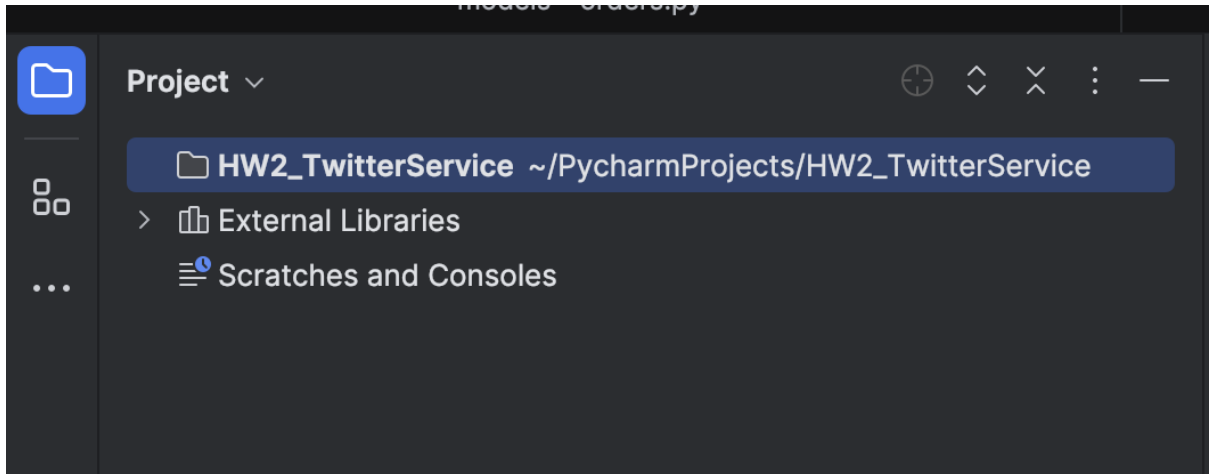


5. Click **Submit** and save the **access token**.

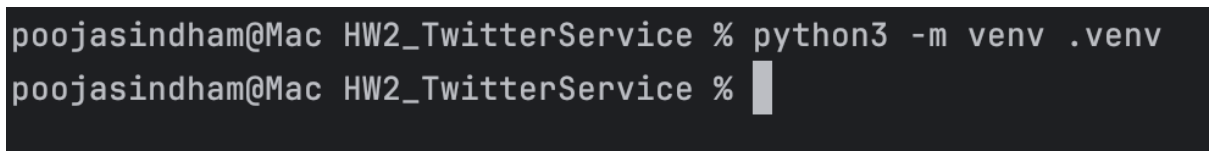


**Create Mastodon Service**

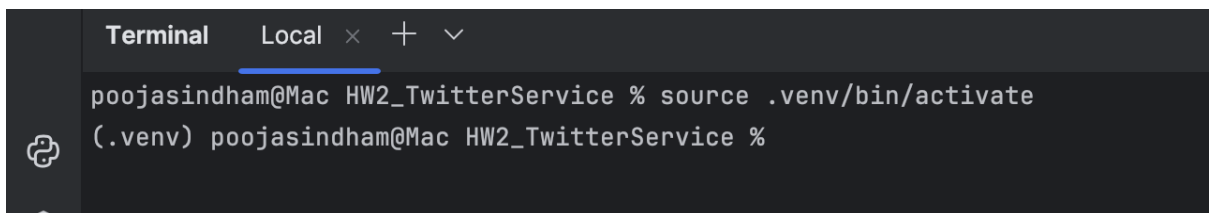
1. Create a new project - HM2\_TwitterService



2. Create virtual environment.

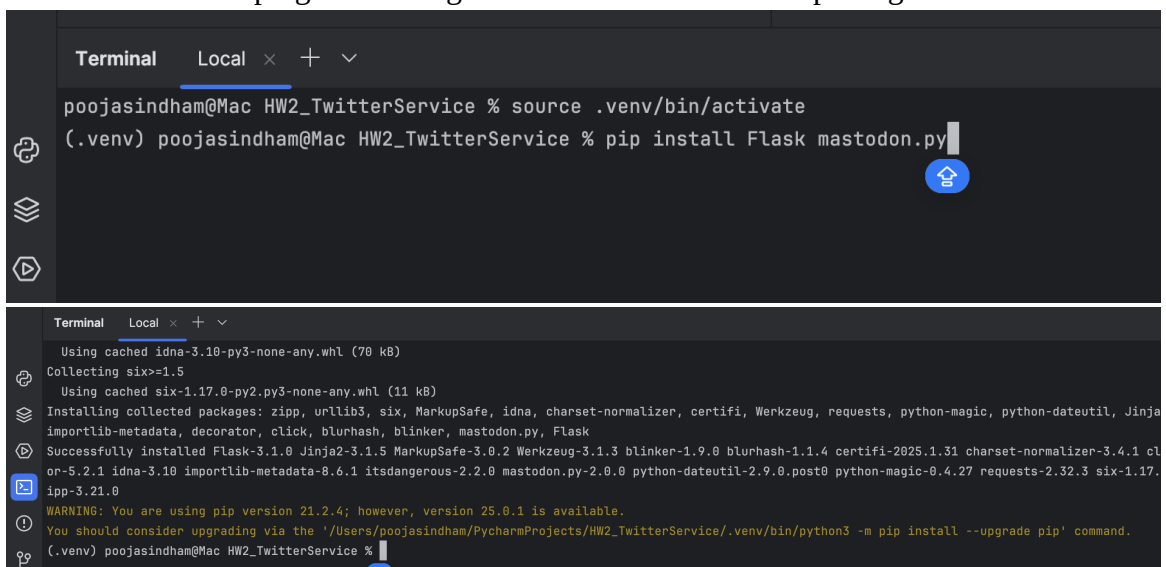


3. Activate the virtual environment.

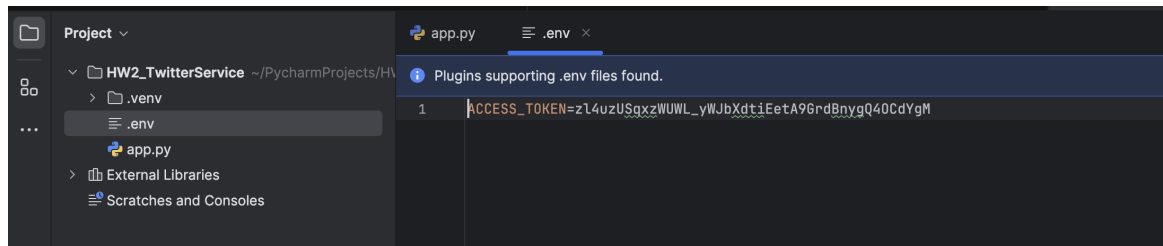


4. Install packages.

Here we are developing API through Flask and Mastodon API package.



5. Create .env file to store Access Token.



6. Set up flask server code to create, retrieve and delete post.  
Create app.py

A. Import the required modules

```
app.py x .env
1 from flask import Flask, render_template, request, jsonify
2 from mastodon import Mastodon, MastodonAPIError, MastodonNetworkError, MastodonUnauthorizedError
3 import os
4 from dotenv import load_dotenv
5 load_dotenv()
6
```

**Flask:** Web framework for handling HTTP requests.

**render\_template:** Renders HTML templates.

**request:** Gets data from the frontend.

**jsonify:** Converts Python dictionaries to JSON format.

**mastodon:** Mastodon API wrapper for Python.

**MastodonAPIError, MastodonNetworkError, MastodonUnauthorizedError:**  
Handles different API errors.

## B. Initialize Flask and Mastodon

```
app.py x .env
6
7 app = Flask(__name__)
8
9 # Initialize Mastodon API
10 mastodon = Mastodon(
11     access_token=os.getenv('ACCESS_TOKEN'),
12     api_base_url="https://mastodon.social"
13 )
```

`app = Flask(__name__)`: Creates a Flask app instance.

`Mastodon(...)`: Connects to the Mastodon API using an access token.

## C. Function for handling errors

```
app.py x .env
14
15 #Function for error handling
16 def handle_error(error_message, status_code=500): 11 usages
17     return jsonify({"error": error_message}), status_code
```

This function will return JSON responses for errors with a message and HTTP status code.

## D. Home Page

```
app.py x .env
18
19 @app.route("/")
20 def home():
21     return render_template("index.html")
22
```

This will render `index.html`.

## E. CreatePost

```
app.py x .env
23 @app.route(rule: "/post", methods=["POST"])
24 def create_post():
25     try:
26         content = request.json.get("content", "")
27         if not content.strip():
28             return handle_error(error_message: "Post content cannot be empty", status_code: 400)
29         post = mastodon.status_post(content)
30         return jsonify({"message": "Post created", "id": post["id"]})
31     except MastodonAPIError as e:
32         return handle_error(f"Mastodon API error: {str(e)}")
33     except MastodonNetworkError:
34         return handle_error(error_message: "Network error. Please check your connection.", status_code: 503)
35     except MastodonUnauthorizedError:
36         return handle_error(error_message: "Invalid API credentials. Please check your access token.", status_code: 401)
37     except Exception as e:
38         return handle_error(f"Unexpected error: {str(e)}")
```

## F. Get Post

```
app.py x <> index.html styles.css JS script.js .env
40 @app.route(rule: "/get/<post_id>", methods=["GET"])
41 def get_post(post_id):
42     try:
43         post = mastodon.status(post_id)
44         return jsonify({"content": post["content"], "created_at": post["created_at"]})
45     except MastodonAPIError as e:
46         return handle_error(f"Failed to retrieve post: {str(e)}")
47     except Exception as e:
48         return handle_error(f"Unexpected error: {str(e)}")
```

## G. Delete Post

```
app.py x <> index.html styles.css JS script.js .env
49
50 @app.route(rule: "/delete/<post_id>", methods=["DELETE"])
51 def delete_post(post_id):
52     try:
53         mastodon.status_delete(post_id)
54         return jsonify({"message": "Post deleted successfully"})
55     except MastodonAPIError as e:
56         return handle_error(f"Failed to delete post: {str(e)}")
57     except Exception as e:
58         return handle_error(f"Unexpected error: {str(e)}")
59
```

## H. Get User Details

```
app.py x  <> index.html  styles.css  JS script.js  .env
59
60 @app.route(rule: "/user", methods=["GET"])
61 def get_user_details():
62     try:
63         account = mastodon.account_verify_credentials()
64         return jsonify({
65             "username": account["username"],
66             "followers": account["followers_count"],
67             "following": account["following_count"],
68             "statuses": account["statuses_count"]
69         })
70     except MastodonAPIError as e:
71         return handle_error(f"Failed to fetch user details: {str(e)}")
72     except Exception as e:
73         return handle_error(f"Unexpected error: {str(e)}")
74
```

## 7. Simple UI to interact with Flask API

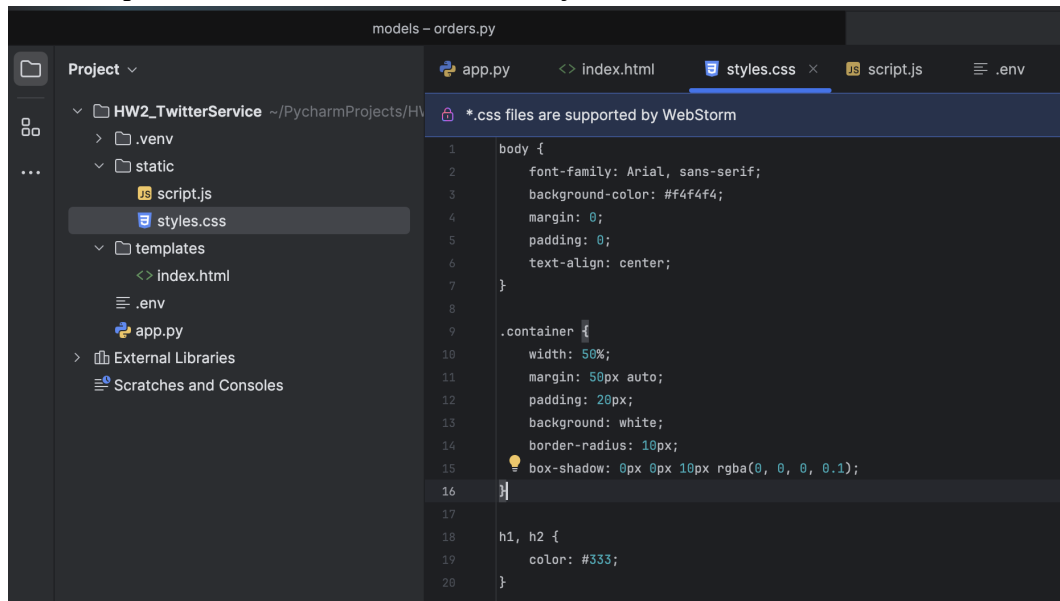
### A. index.html in templates directory

```
Project v
  HW2_TwitterService ~/PycharmProjects/HW2_TwitterService
  .venv
  static
  script.js
  styles.css
  templates
    index.html
  .env
  app.py
  External Libraries
  Scratches and Consoles

app.py  <> index.html  styles.css  JS script.js  .env
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Mastodon API Integration</title>
6   <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
7 </head>
8 <body>
9   <div class="container">
10    <h1>Mastodon API Integration</h1>
11
12    <div id="messageBox"></div>
13
14    <h2>Create Post</h2>
15    <textarea id="postContent" placeholder="Write something..."></textarea>
16    <button onclick="createPost()">Post</button>
17    <input type="text" id="postId" placeholder="Enter Post ID">
18    <button onclick="getPost()">Fetch Post</button>
19    <button onclick="deletePost()">Delete Post</button>
20    <h2>User Details</h2>
21    <button onclick="getUserDetails()">Get User Info</button>
22  </div>
23  <script src="{{ url_for('static', filename='script.js') }}"></script>
24 </body>
25 </html>
26
```

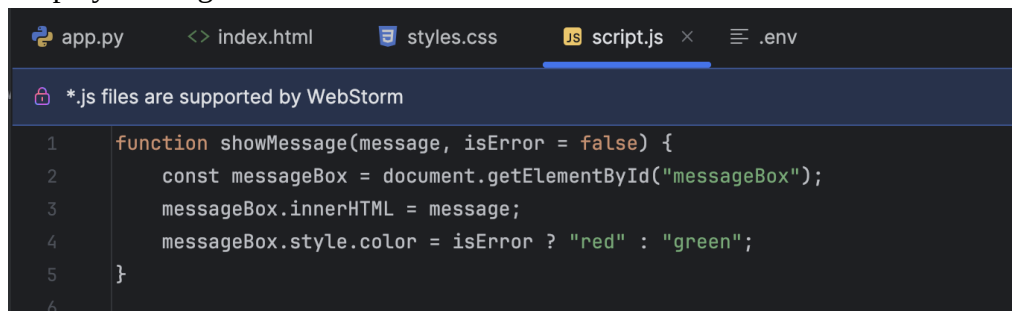


B. Add Simple CSS for look and feel of UI. styles.css files under static folder.

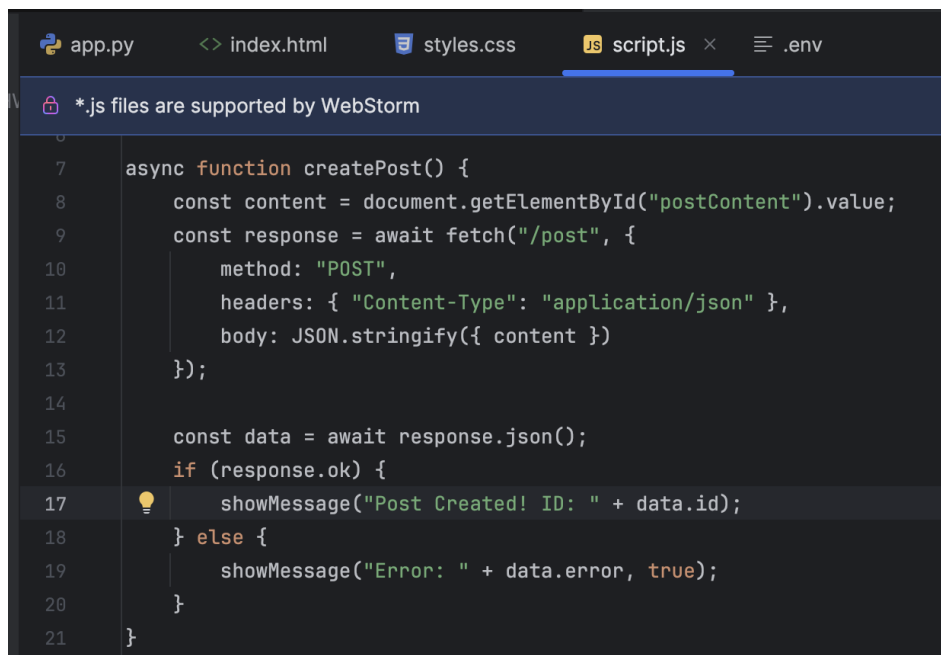


C. JavaScript to handle API request and error handling

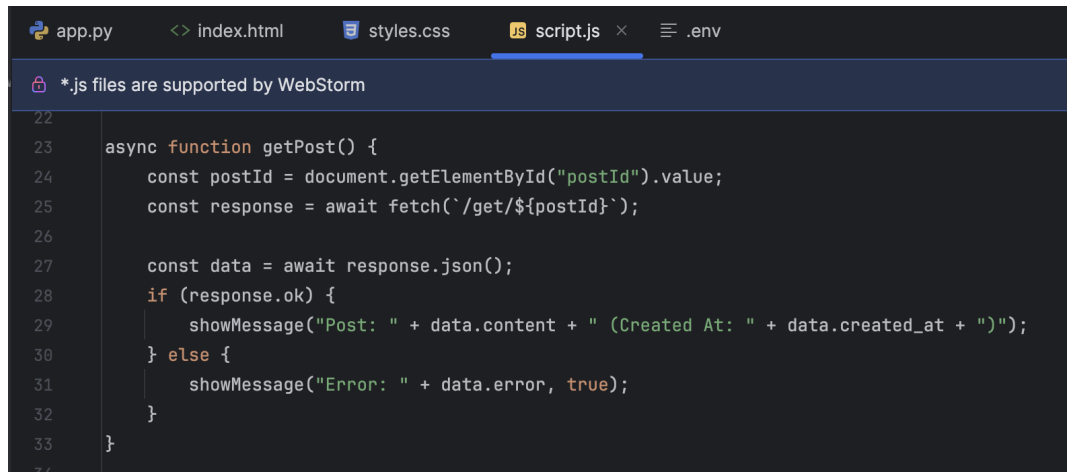
- Display Message



- Create Post



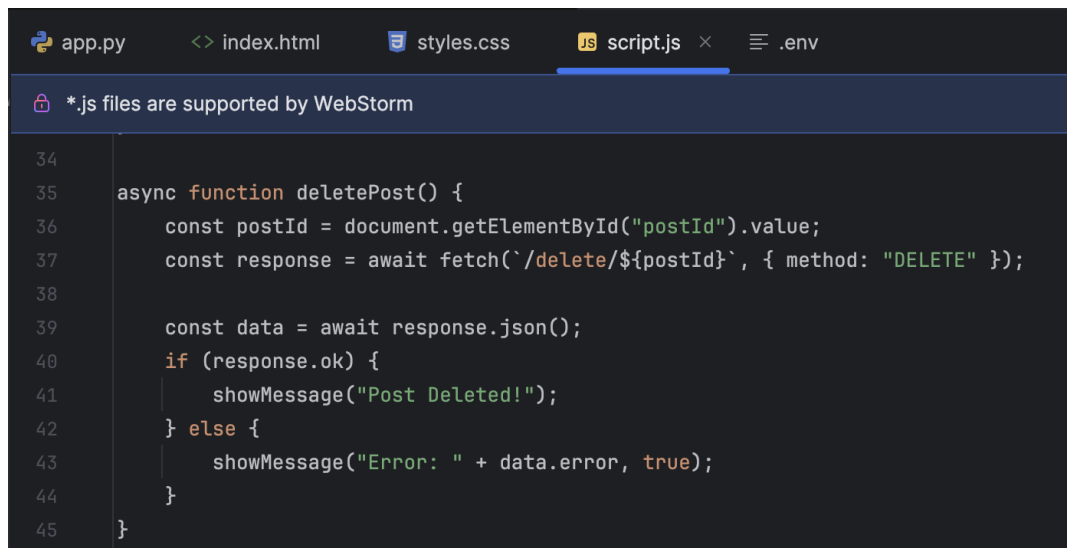
- Get Post



```
app.py  <> index.html  styles.css  JS script.js  .env
*.js files are supported by WebStorm

22
23  async function getPost() {
24      const postId = document.getElementById("postId").value;
25      const response = await fetch(`/get/${postId}`);
26
27      const data = await response.json();
28      if (response.ok) {
29          showMessage("Post: " + data.content + " (Created At: " + data.created_at + ")");
30      } else {
31          showMessage("Error: " + data.error, true);
32      }
33  }
34
```

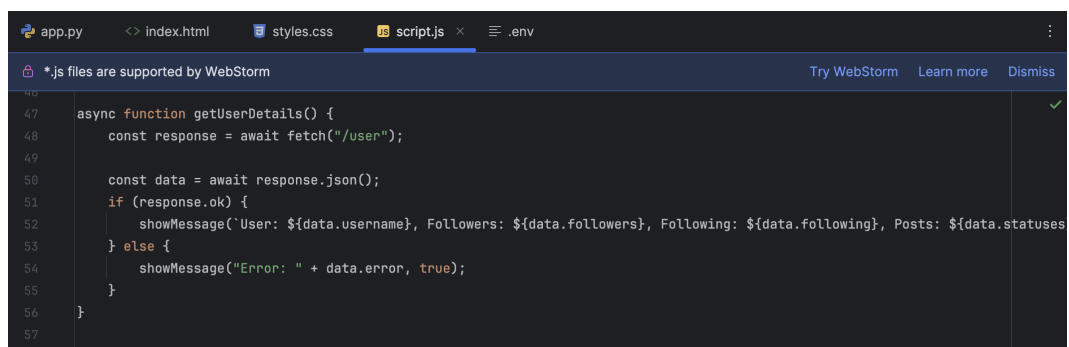
- Delete Post



```
app.py  <> index.html  styles.css  JS script.js  .env
*.js files are supported by WebStorm

34
35  async function deletePost() {
36      const postId = document.getElementById("postId").value;
37      const response = await fetch(`/delete/${postId}`, { method: "DELETE" });
38
39      const data = await response.json();
40      if (response.ok) {
41          showMessage("Post Deleted!");
42      } else {
43          showMessage("Error: " + data.error, true);
44      }
45  }
46
```

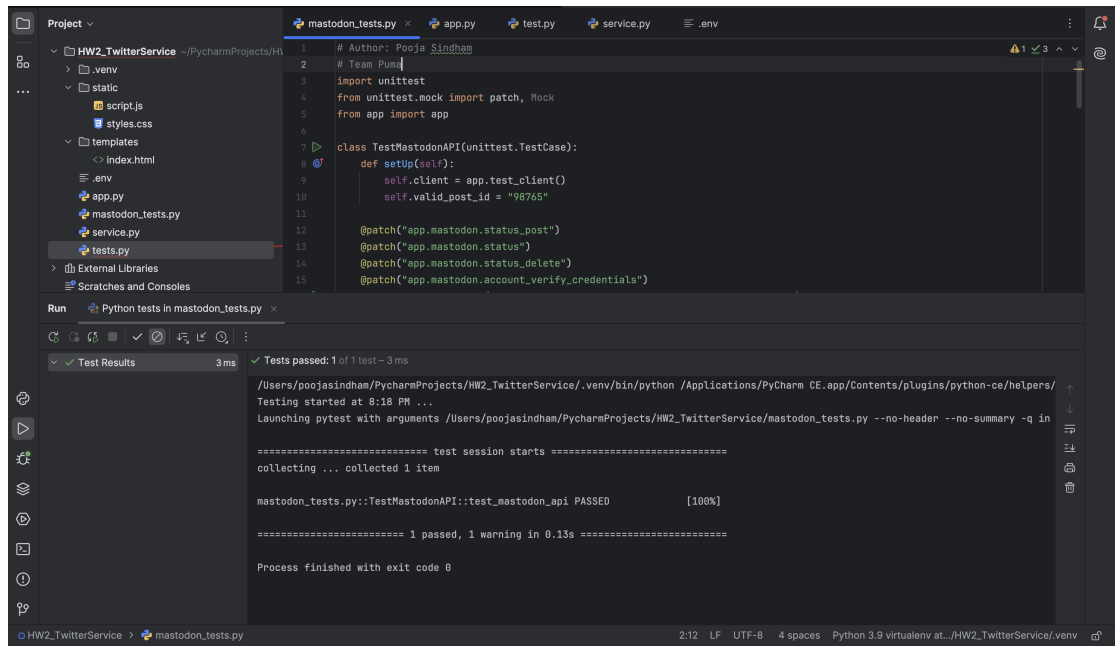
- Get User Details



```
app.py  <> index.html  styles.css  JS script.js  .env
*.js files are supported by WebStorm  Try WebStorm  Learn more  Dismiss

47  async function getUserDetails() {
48      const response = await fetch("/user");
49
50      const data = await response.json();
51      if (response.ok) {
52          showMessage(`User: ${data.username}, Followers: ${data.followers}, Following: ${data.following}, Posts: ${data.statuses}`);
53      } else {
54          showMessage("Error: " + data.error, true);
55      }
56  }
57
```

## D. Unit tests.



The screenshot shows the PyCharm IDE interface. The top pane displays the code for `mastodon_tests.py`, which includes imports for `unittest` and `unittest.mock`, and a test class `TestMastodonAPI` with a `setUp` method and several `@patch` decorators. The bottom pane shows the test results, indicating that the test passed successfully.

```
1 # Author: Pooja Sindham
2 # Team: Puma
3 import unittest
4 from unittest.mock import patch, Mock
5 from app import app
6
7 class TestMastodonAPI(unittest.TestCase):
8     def setUp(self):
9         self.client = app.test_client()
10        self.valid_post_id = "98765"
11
12        @patch("app.mastodon.status_post")
13        @patch("app.mastodon.status")
14        @patch("app.mastodon.status_delete")
15        @patch("app.mastodon.account_verify_credentials")
```

Run Python tests in mastodon\_tests.py

Test Results 3 ms ✓ Tests passed: 1 of 1 test - 3 ms

/Users/poojasindham/PycharmProjects/HW2\_TwitterService/.venv/bin/python /Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/Testing started at 8:18 PM ...  
Launching pytest with arguments /Users/poojasindham/PycharmProjects/HW2\_TwitterService/mastodon\_tests.py --no-header --no-summary -q in

===== test session starts =====  
collecting ... collected 1 item

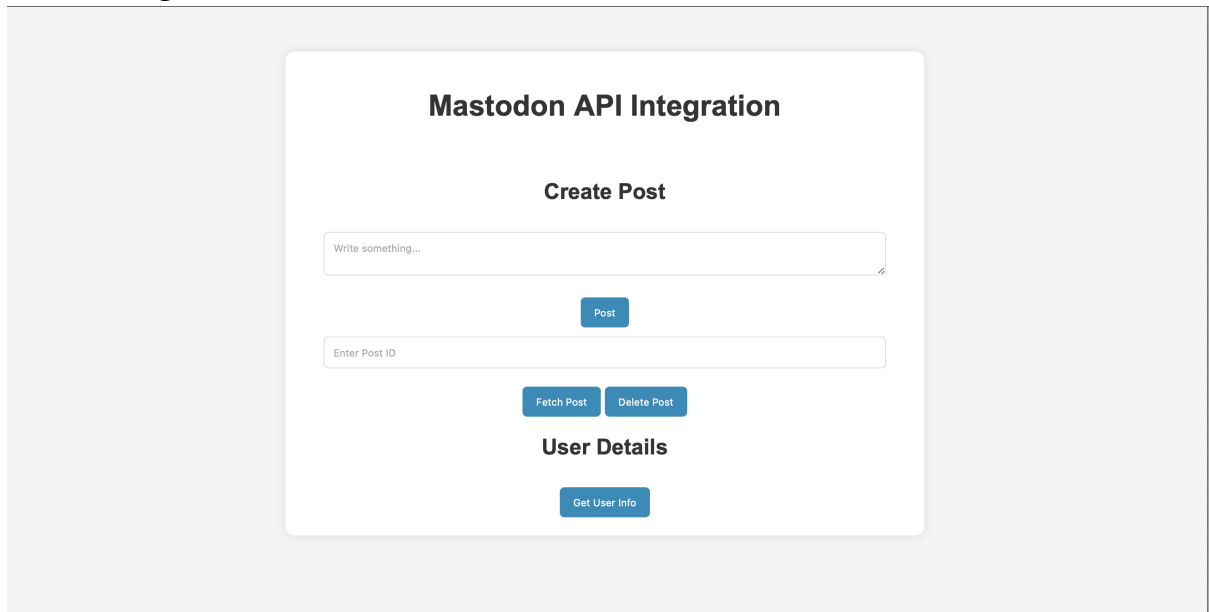
mastodon\_tests.py::TestMastodonAPI::test\_mastodon\_api PASSED [100%]

===== 1 passed, 1 warning in 0.13s =====

Process finished with exit code 0

## UI (Frontend)

### 1. Home Page



The screenshot shows a web application titled "Mastodon API Integration". It features two main sections: "Create Post" and "User Details".

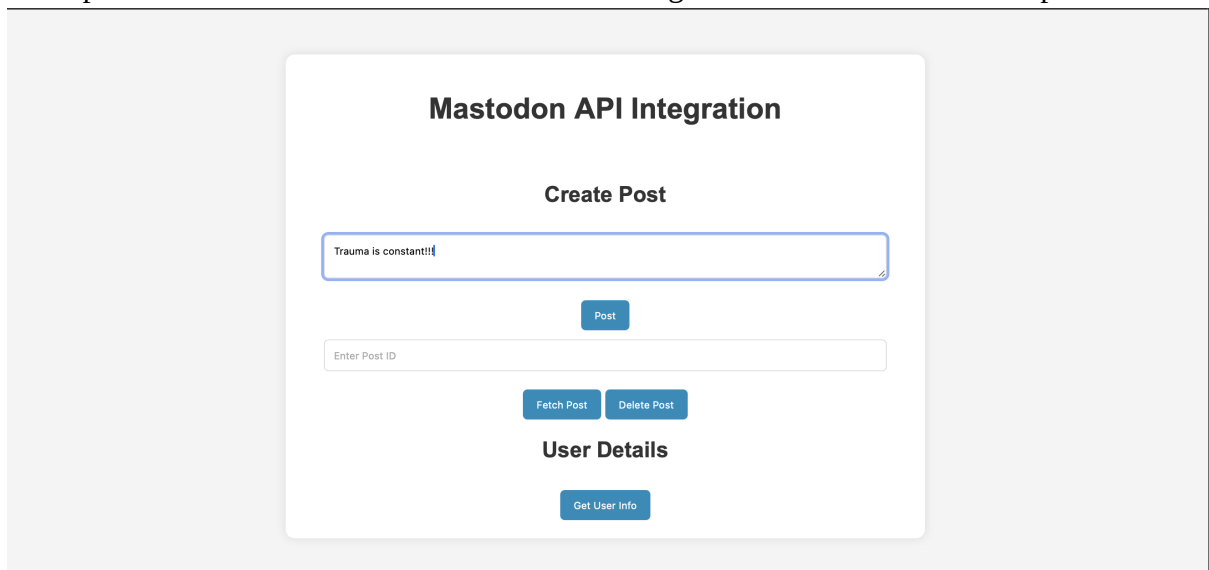
**Create Post**

- A text input field with the placeholder text "Write something...".
- A blue "Post" button.
- A text input field with the placeholder text "Enter Post ID".
- Two blue buttons: "Fetch Post" and "Delete Post".

**User Details**

- A blue "Get User Info" button.

### 2. To post a text on mastodon social write something in the textbox and click on post.



This screenshot shows the same web application as the first one, but with the text "Trauma is constant!!!" entered into the "Write something..." text input field. The "Post" button is highlighted with a blue border, indicating it is the next step in the process.

**Create Post**

- The text input field now contains "Trauma is constant!!!".
- The "Post" button is highlighted with a blue border.
- The "Enter Post ID" field and "Fetch Post" / "Delete Post" buttons remain unchanged.

**User Details**

- The "Get User Info" button remains unchanged.

3. Once you click on post. You will see the post ID that got posted.

### Mastodon API Integration

Post Created! ID: 114089494719624758

#### Create Post

Trauma is constant!!!

Post

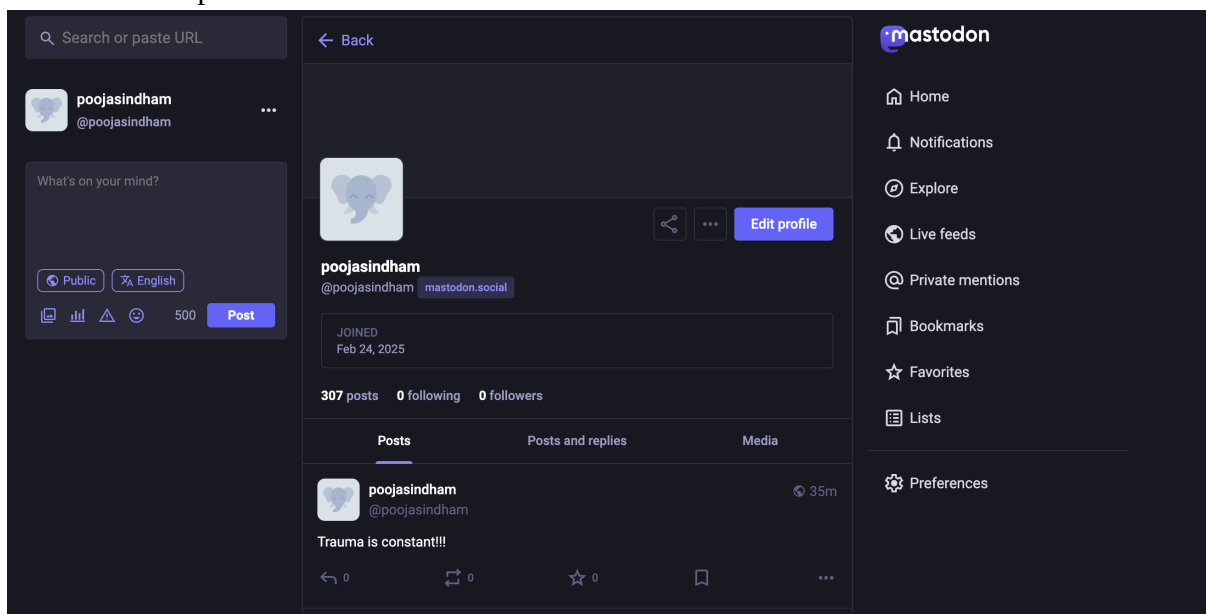
Enter Post ID

Fetch PostDelete Post

#### User Details

Get User Info

You can see it posted on mastodon social.



4. Now you can use this post id to retrieve or delete the post.

5. Retrieve the post. Give the post id in the second text box and click on Fetch post.

### Mastodon API Integration

**Post:**  
Trauma is constant!!!  
(Created At: Sat, 01 Mar 2025 22:18:54 GMT)

#### Create Post

Post

Fetch PostDelete Post

#### User Details

Get User Info

6. Delete the post. Give the post id in second text box and click delete post.

### Mastodon API Integration

**Post Deleted!**

#### Create Post

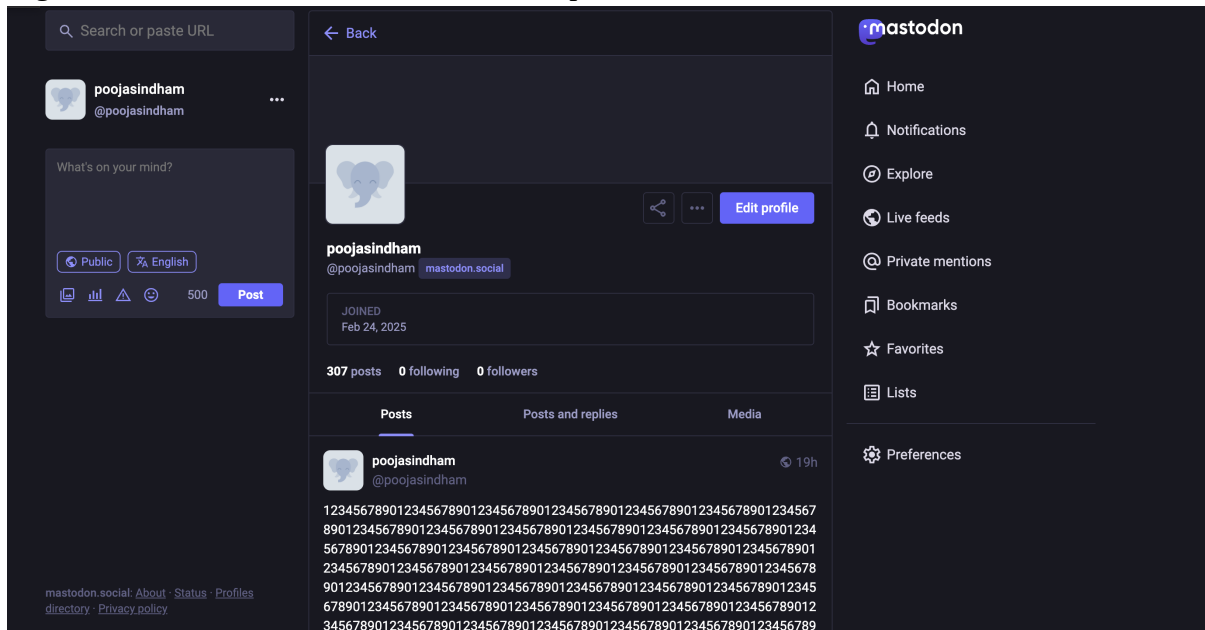
Post

Fetch PostDelete Post

#### User Details

Get User Info

It got reflected in mastodon API, where the post will be deleted.



7. Get User Info will fetch the user details like followers, following and number of posts.

