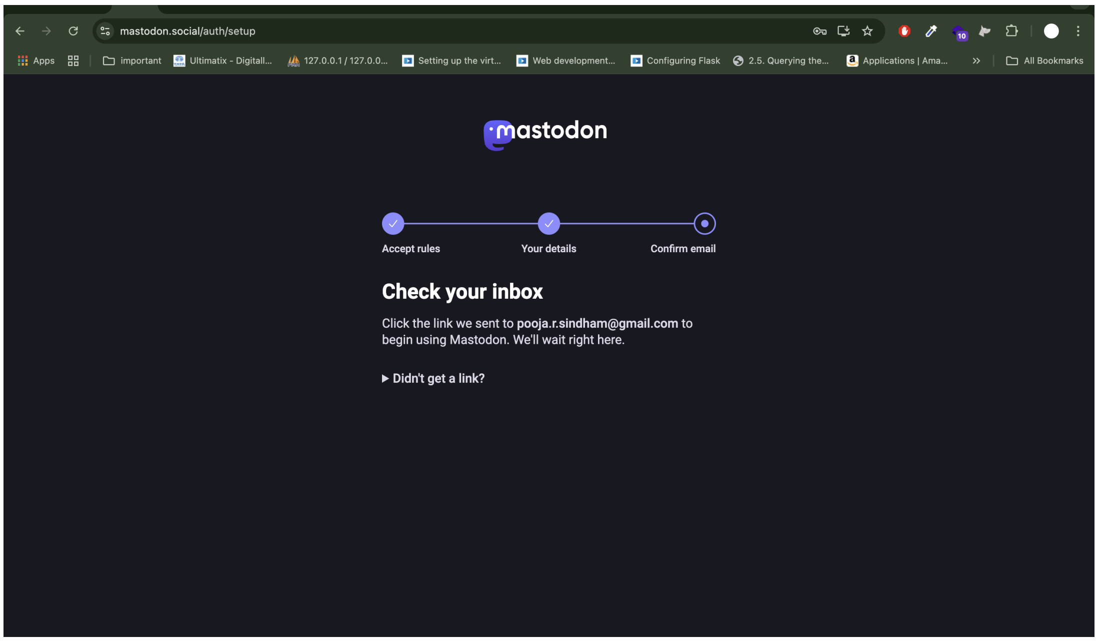
Homework #2 – Twitter Service

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

Get Access token for Mostodon Social

1. Signup for mastodon social.



mastodon.social/auth/setup

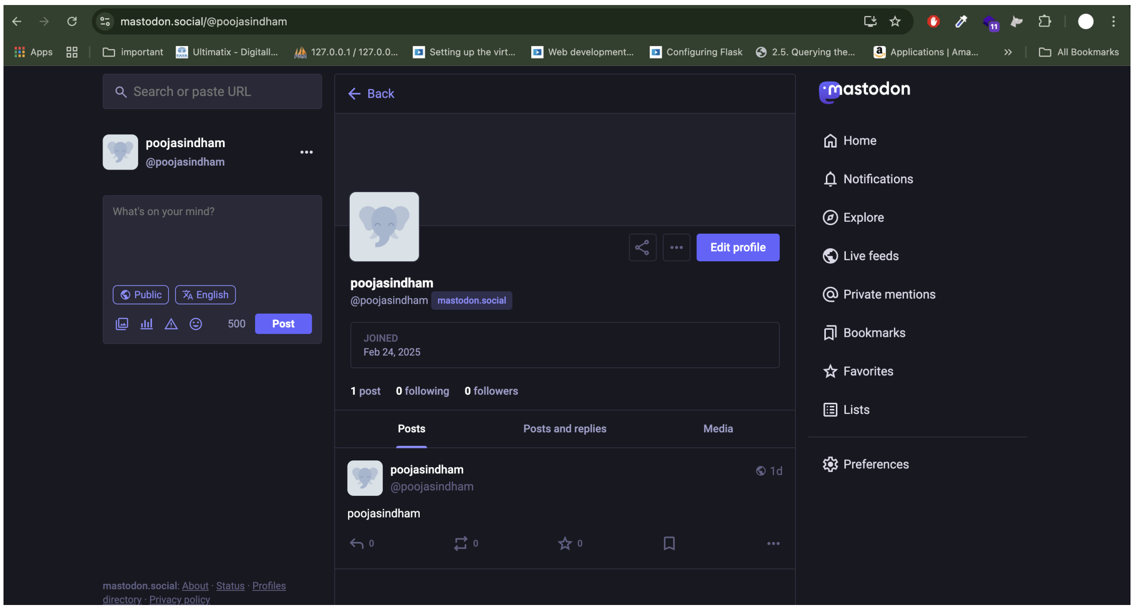
Accept rules · Your details · Confirm email

Check your inbox

Click the link we sent to **pooja.r.sindham@gmail.com** to begin using Mastodon. We'll wait right here.

▸ Didn't get a link?

2. Login to your account and go to preferences.



mastodon.social/@poojasindham

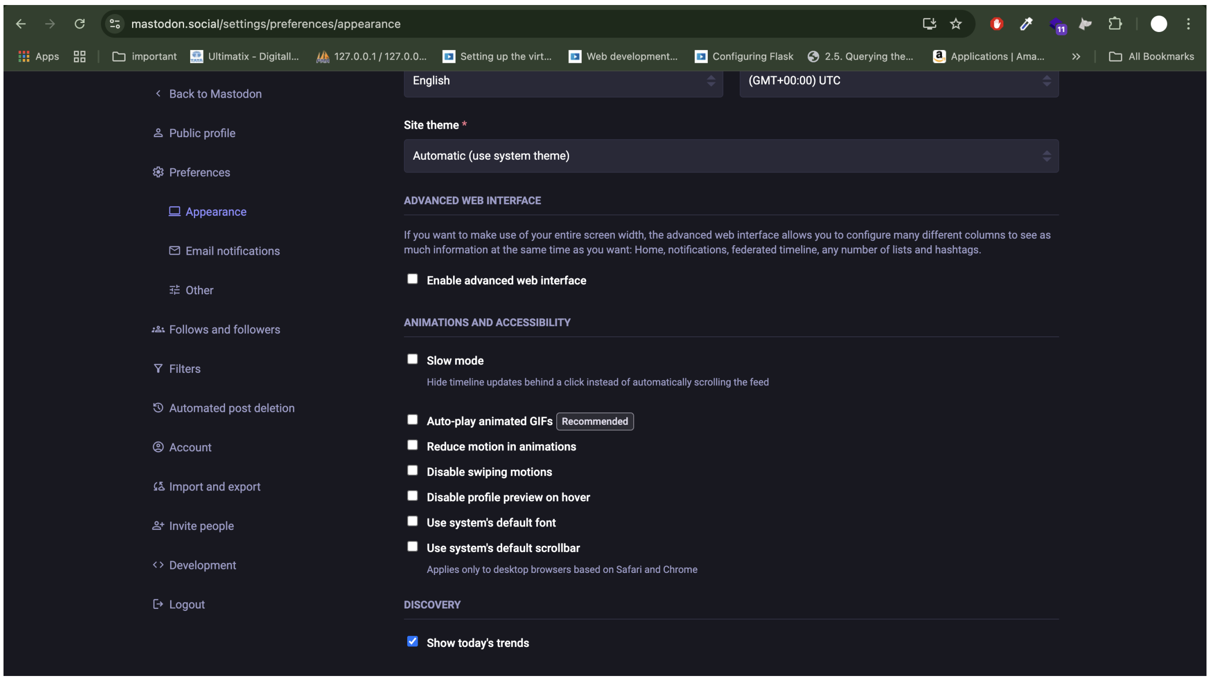
poojasindham
@poojasindham mastodon.social

JOINED
Feb 24, 2025

1 post 0 following 0 followers

poojasindham @poojasindham 1d
poojasindham

3. Navigate to Preferences > Development.


mastodon.social/settings/preferences/appearance

English | (GMT+00:00) UTC

Site theme *

Automatic (use system theme)

ADVANCED WEB INTERFACE

If you want to make use of your entire screen width, the advanced web interface allows you to configure many different columns to see as much information at the same time as you want: Home, notifications, federated timeline, any number of lists and hashtags.

- [ ] Enable advanced web interface

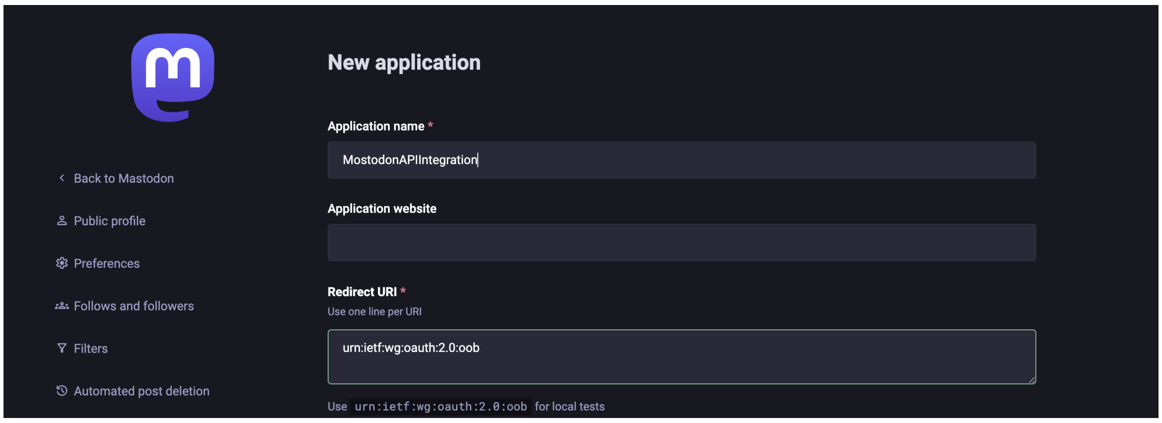
ANIMATIONS AND ACCESSIBILITY

- [ ] Slow mode
  Hide timeline updates behind a click instead of automatically scrolling the feed
- [ ] Auto-play animated GIFs Recommended
- [ ] Reduce motion in animations
- [ ] Disable swiping motions
- [ ] Disable profile preview on hover
- [ ] Use system's default font
- [ ] Use system's default scrollbar
  Applies only to desktop browsers based on Safari and Chrome

DISCOVERY

- [x] Show today's trends

4. Create new application.



New application

Application name *

MostodonAPIIntegration

Application website

Redirect URI *

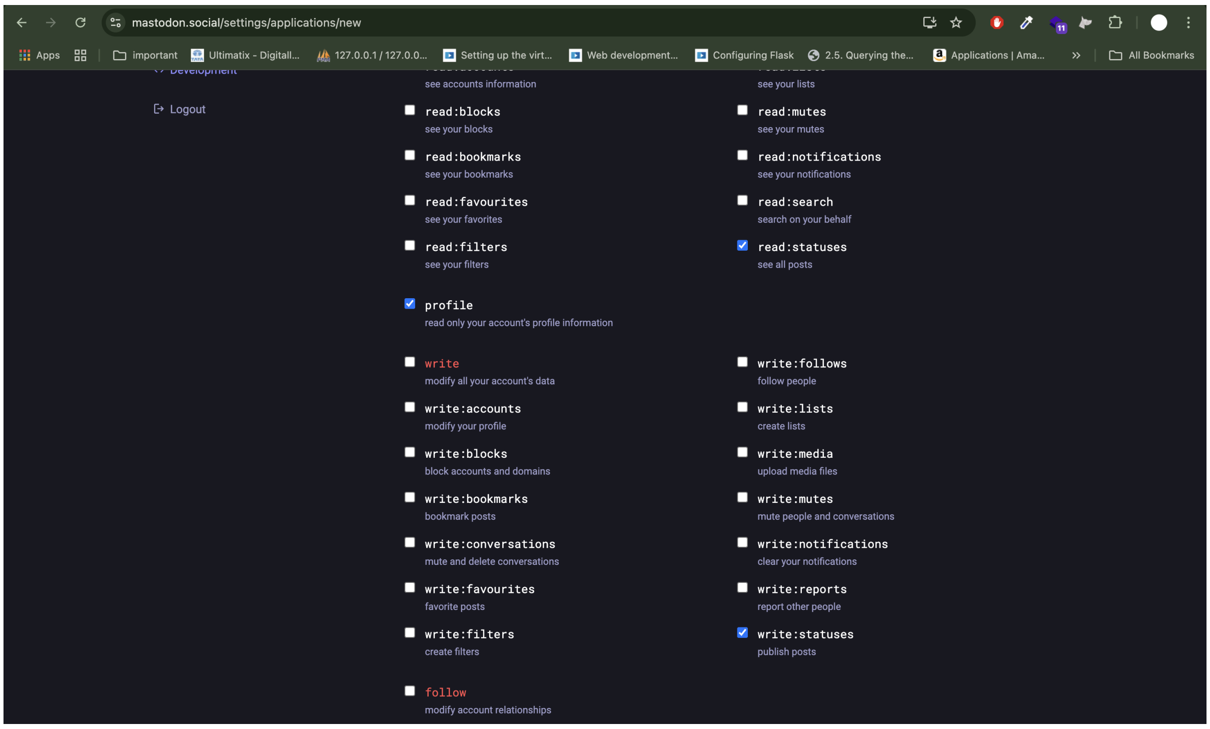
Use one line per URI

urn:ietf:wg:oauth:2.0:oob

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

Select the following options

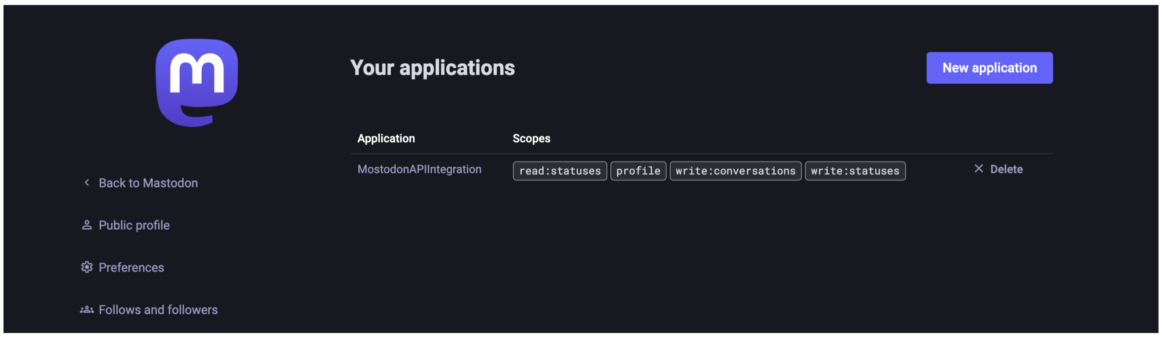
- *read:statuses (Retrieve posts)*
- *write:statuses (Create and **delete** posts)*


mastodon.social/settings/applications/new

Logout

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

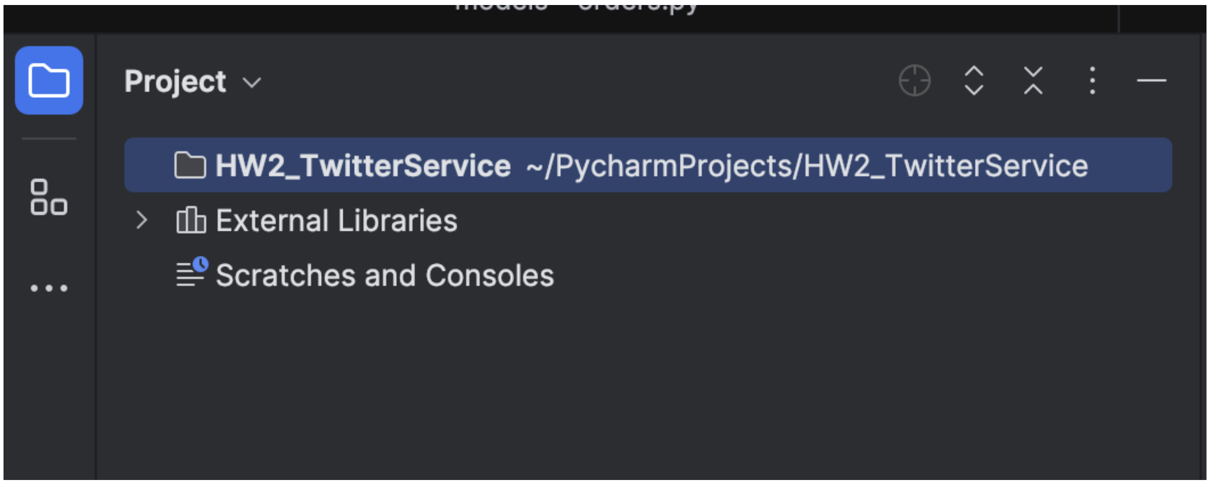
5. Click **Submit** and save the **access token**.


Your applications

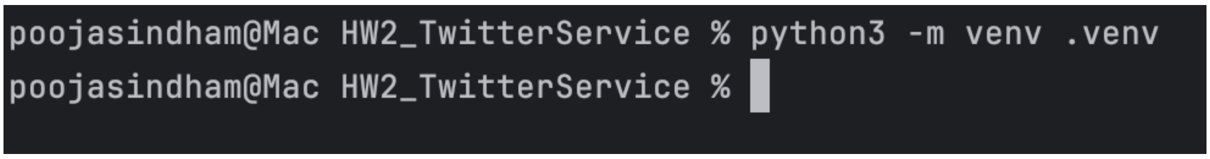
Application	Scopes
MostodonAPIIntegration	read:statuses profile write:conversations write:statuses

Create Mastodon Service

1. Create a new project - HM2_TwitterService

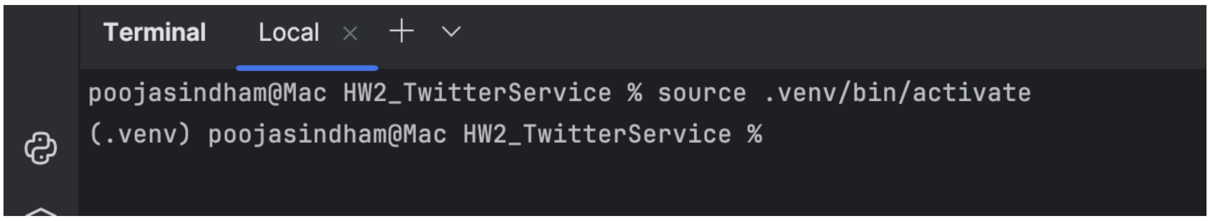


2. Create virtual environment.



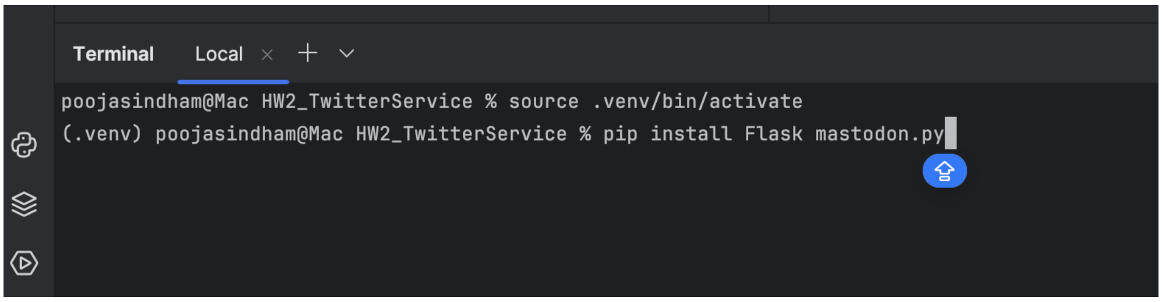
```
poojasindham@Mac HW2_TwitterService % python3 -m venv .venv
poojasindham@Mac HW2_TwitterService %
```

3. Activate the virtual environment.

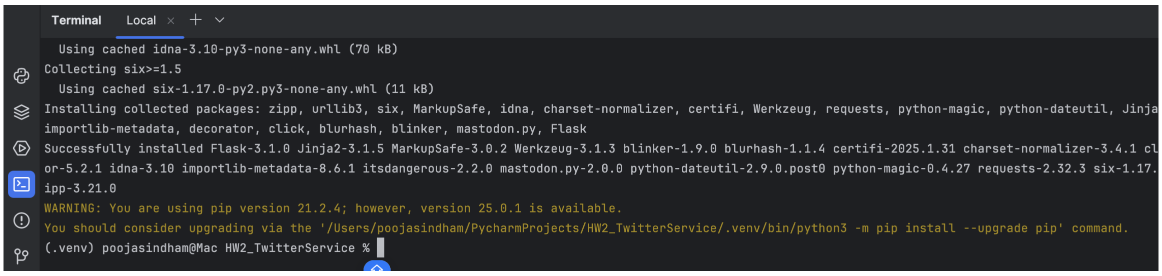


```
poojasindham@Mac HW2_TwitterService % source .venv/bin/activate
(.venv) poojasindham@Mac HW2_TwitterService %
```

4. Install packages.
   Here we are developing API through Flask and Mastodon API package.

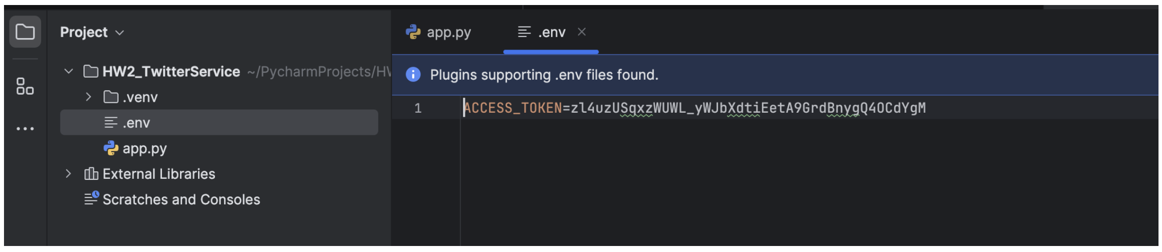


```
poojasindham@Mac HW2_TwitterService % source .venv/bin/activate
(.venv) poojasindham@Mac HW2_TwitterService % pip install Flask mastodon.py
```



```
  Using cached idna-3.10-py3-none-any.whl (70 kB)
Collecting six>=1.5
  Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: zipp, urllib3, six, MarkupSafe, idna, charset-normalizer, certifi, Werkzeug, requests, python-magic, python-dateutil, Jinja
importlib-metadata, decorator, click, blurhash, blinker, mastodon.py, Flask
Successfully installed Flask-3.1.0 Jinja2-3.1.5 MarkupSafe-3.0.2 Werkzeug-3.1.3 blinker-1.9.0 blurhash-1.1.4 certifi-2025.1.31 charset-normalizer-3.4.1 cl
or-5.2.1 idna-3.10 importlib-metadata-8.6.1 itsdangerous-2.2.0 mastodon.py-2.0.0 python-dateutil-2.9.0.post0 python-magic-0.4.27 requests-2.32.3 six-1.17.
ipp-3.21.0
WARNING: You are using pip version 21.2.4; however, version 25.0.1 is available.
You should consider upgrading via the '/Users/poojasindham/PycharmProjects/HW2_TwitterService/.venv/bin/python3 -m pip install --upgrade pip' command.
(.venv) poojasindham@Mac HW2_TwitterService %
```

5. Create .env file to store Access Token.

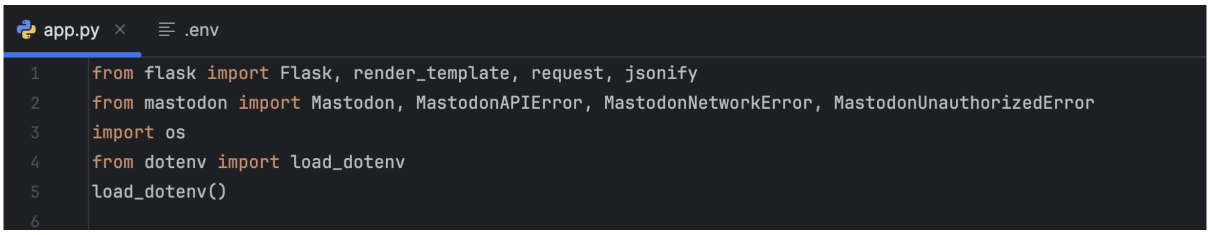


Plugins supporting .env files found.

```
ACCESS_TOKEN=zl4uzUSqxzWUWL_yWJbXdtiEetA9GrdBnygQ4OCdYgM
```

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

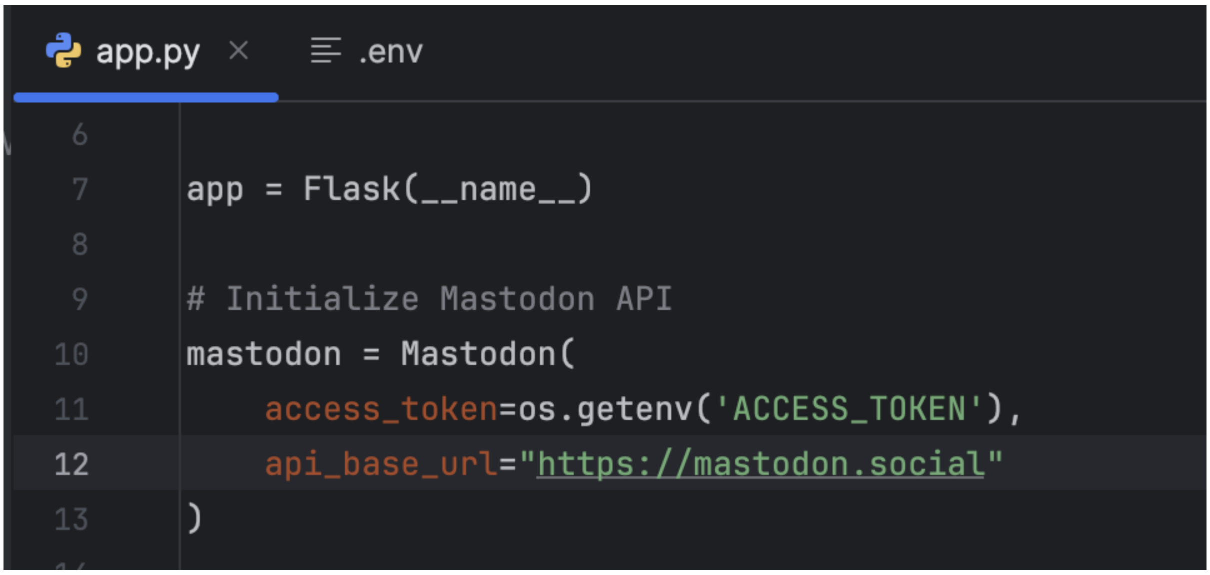
   A. Import the required modules



```python
from flask import Flask, render_template, request, jsonify
from mastodon import Mastodon, MastodonAPIError, MastodonNetworkError, MastodonUnauthorizedError
import os
from dotenv import load_dotenv
load_dotenv()
```

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.

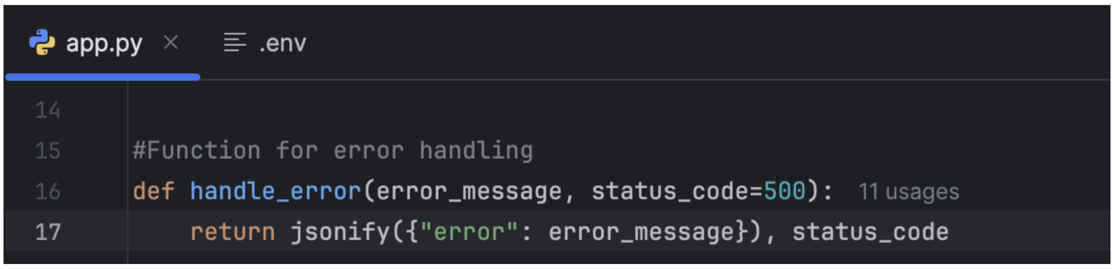
B. Initialize Flask and Mastodon



```
app.py    .env

app = Flask(__name__)

# Initialize Mastodon API
mastodon = Mastodon(
    access_token=os.getenv('ACCESS_TOKEN'),
    api_base_url="https://mastodon.social"
)
```

app = Flask(__name__): Creates a Flask app instance.
Mastodon(...): Connects to the Mastodon API using an access token.

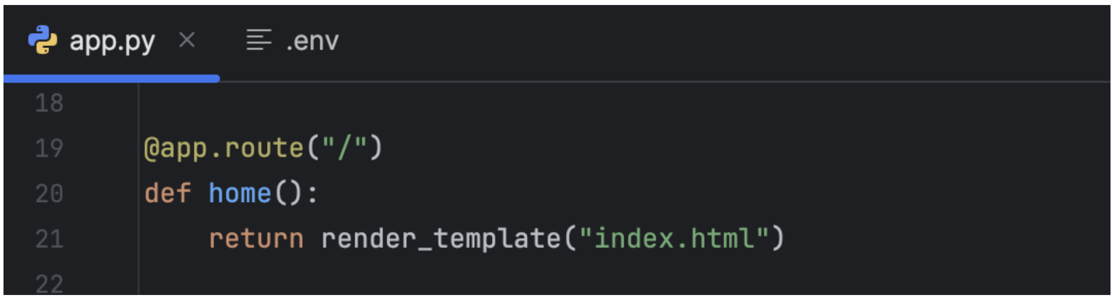
C. Function for handling errors



```
app.py    .env

#Function for error handling
def handle_error(error_message, status_code=500):  11 usages
    return jsonify({"error": error_message}), status_code
```

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

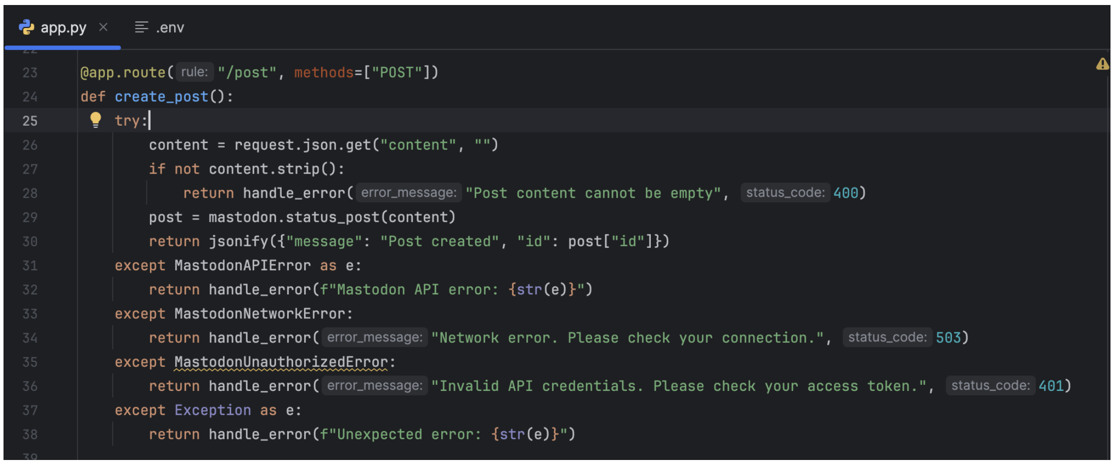
D. Home Page



```
app.py    .env

@app.route("/")
def home():
    return render_template("index.html")
```

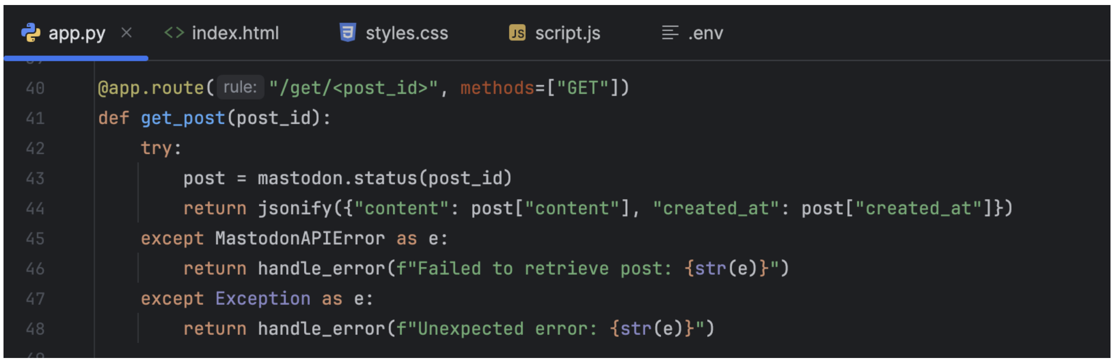
This will render index.html.

E. CreatePost



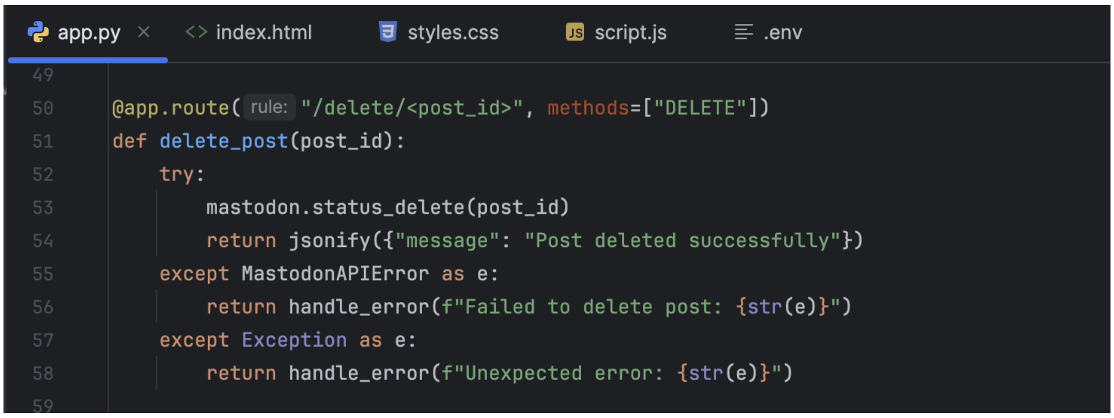
```
@app.route( rule: "/post", methods=["POST"])
def create_post():
    try:
        content = request.json.get("content", "")
        if not content.strip():
            return handle_error( error_message: "Post content cannot be empty", status_code: 400)
        post = mastodon.status_post(content)
        return jsonify({"message": "Post created", "id": post["id"]})
    except MastodonAPIError as e:
        return handle_error(f"Mastodon API error: {str(e)}")
    except MastodonNetworkError:
        return handle_error( error_message: "Network error. Please check your connection.", status_code: 503)
    except MastodonUnauthorizedError:
        return handle_error( error_message: "Invalid API credentials. Please check your access token.", status_code: 401)
    except Exception as e:
        return handle_error(f"Unexpected error: {str(e)}")
```

F. Get Post



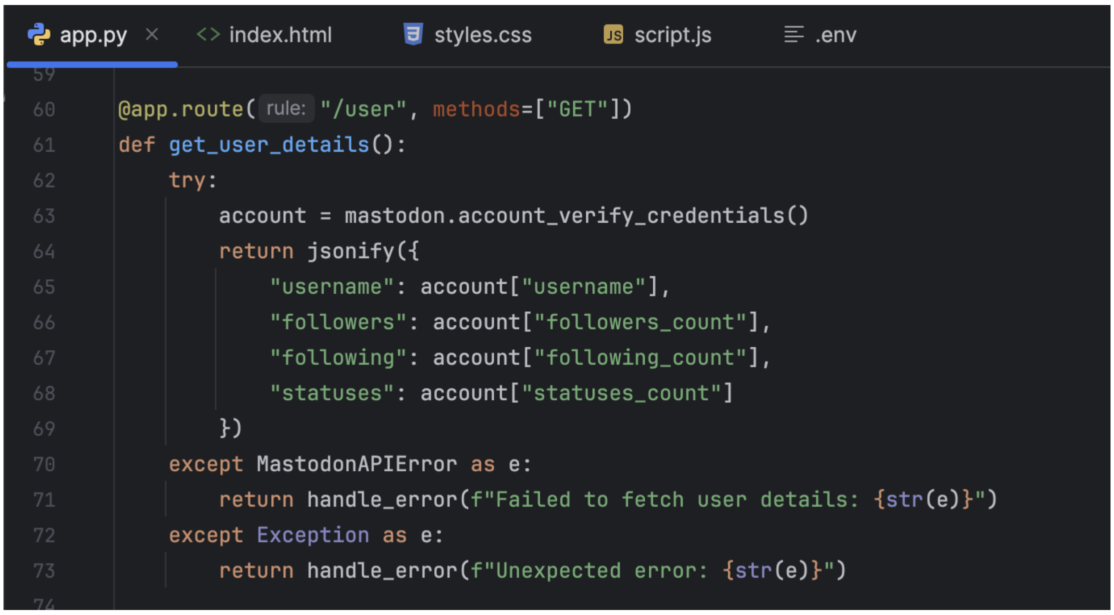
```
@app.route( rule: "/get/<post_id>", methods=["GET"])
def get_post(post_id):
    try:
        post = mastodon.status(post_id)
        return jsonify({"content": post["content"], "created_at": post["created_at"]})
    except MastodonAPIError as e:
        return handle_error(f"Failed to retrieve post: {str(e)}")
    except Exception as e:
        return handle_error(f"Unexpected error: {str(e)}")
```

G. Delete Post



```
@app.route( rule: "/delete/<post_id>", methods=["DELETE"])
def delete_post(post_id):
    try:
        mastodon.status_delete(post_id)
        return jsonify({"message": "Post deleted successfully"})
    except MastodonAPIError as e:
        return handle_error(f"Failed to delete post: {str(e)}")
    except Exception as e:
        return handle_error(f"Unexpected error: {str(e)}")
```

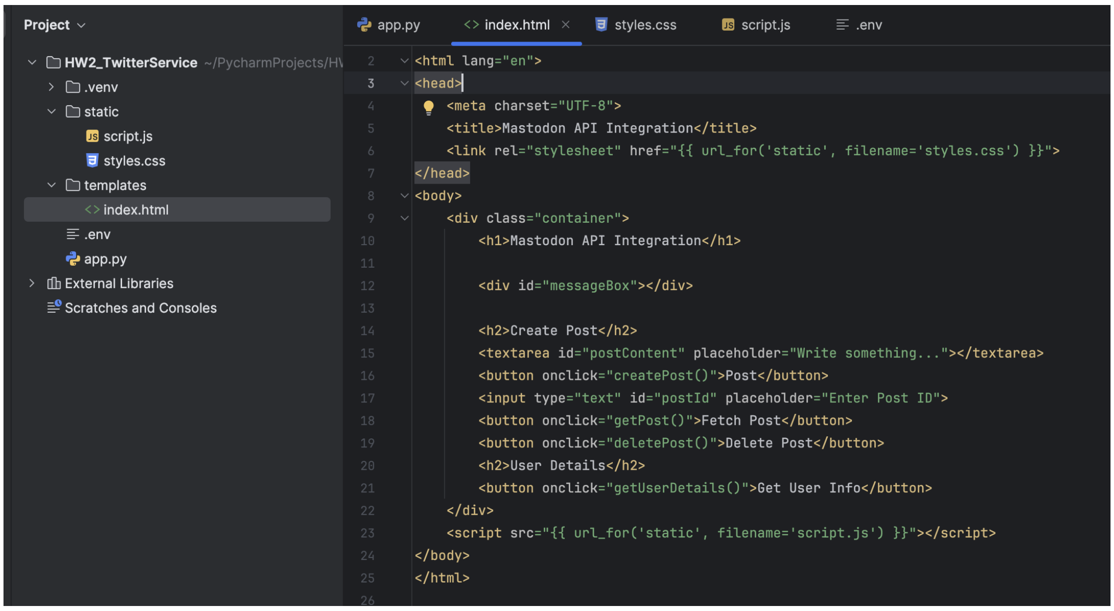
H. Get User Details



```python
@app.route( rule: "/user", methods=["GET"])
def get_user_details():
    try:
        account = mastodon.account_verify_credentials()
        return jsonify({
            "username": account["username"],
            "followers": account["followers_count"],
            "following": account["following_count"],
            "statuses": account["statuses_count"]
        })
    except MastodonAPIError as e:
        return handle_error(f"Failed to fetch user details: {str(e)}")
    except Exception as e:
        return handle_error(f"Unexpected error: {str(e)}")
```

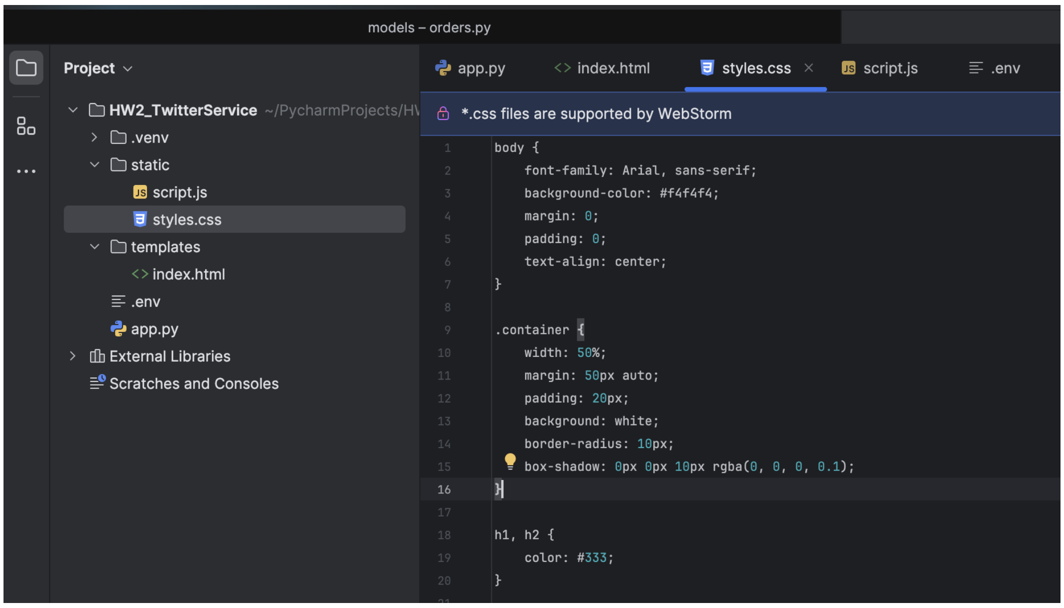
7. Simple UI to interact with Flask API

A. index.html in templates directory



```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Mastodon API Integration</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Mastodon API Integration</h1>

        <div id="messageBox"></div>

        <h2>Create Post</h2>
        <textarea id="postContent" placeholder="Write something..."></textarea>
        <button onclick="createPost()">Post</button>
        <input type="text" id="postId" placeholder="Enter Post ID">
        <button onclick="getPost()">Fetch Post</button>
        <button onclick="deletePost()">Delete Post</button>
        <h2>User Details</h2>
        <button onclick="getUserDetails()">Get User Info</button>
    </div>
    <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

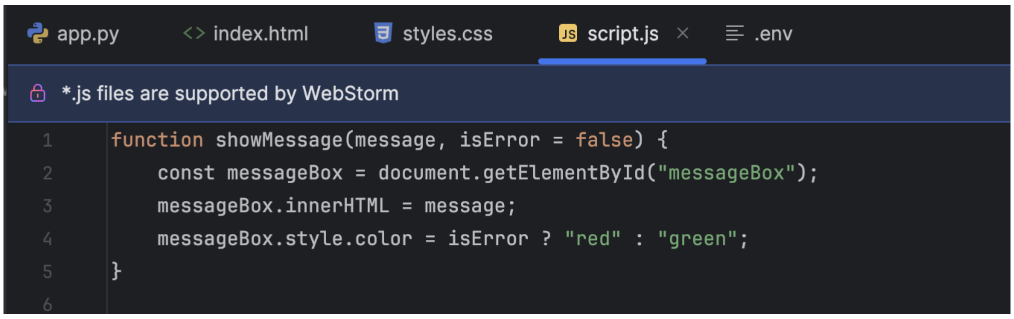
B. Add Simple CSS for look and feel of UI. styles.css files under static folder.



```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    text-align: center;
}

.container {
    width: 50%;
    margin: 50px auto;
    padding: 20px;
    background: white;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

h1, h2 {
    color: #333;
}
```

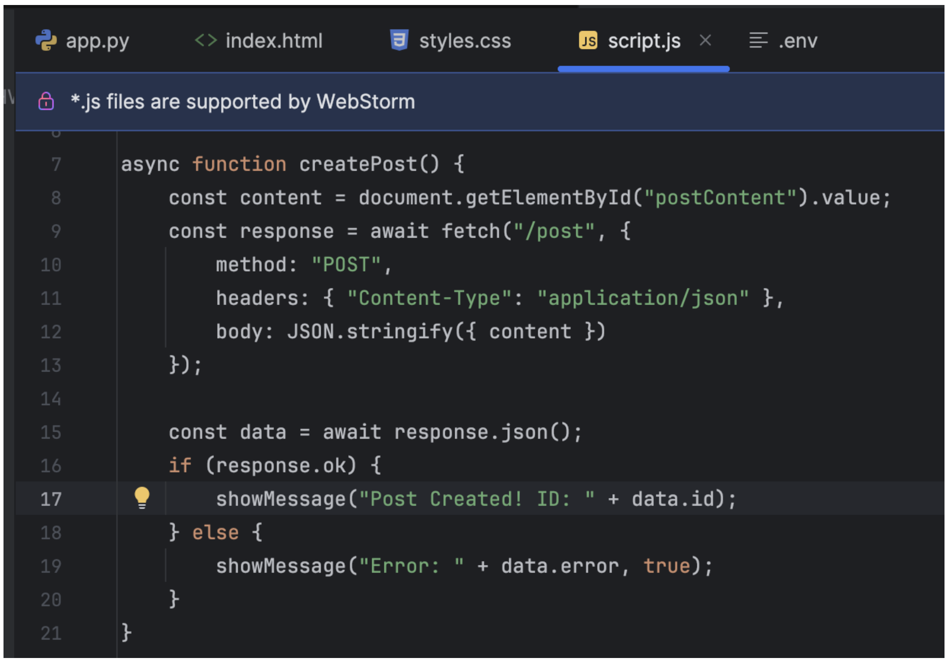
C. JavaScript to handle API request and error handling

- Display Message



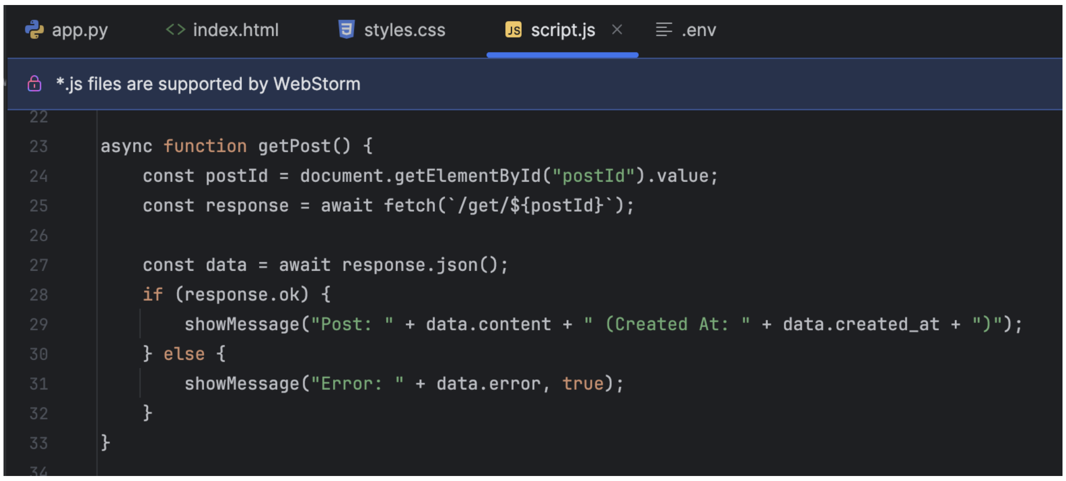
```javascript
function showMessage(message, isError = false) {
    const messageBox = document.getElementById("messageBox");
    messageBox.innerHTML = message;
    messageBox.style.color = isError ? "red" : "green";
}
```

- Create Post



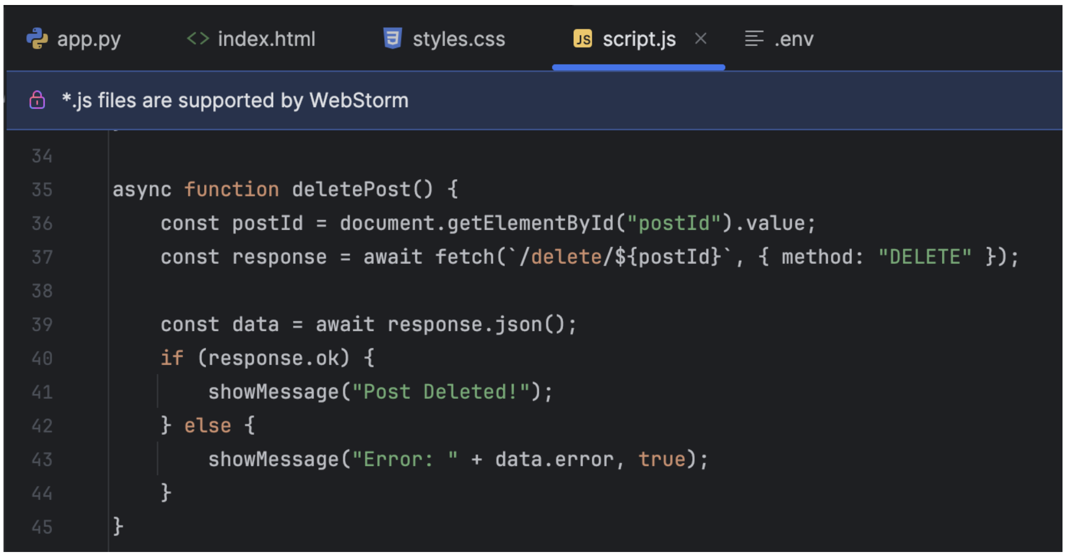
```javascript
async function createPost() {
    const content = document.getElementById("postContent").value;
    const response = await fetch("/post", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ content })
    });

    const data = await response.json();
    if (response.ok) {
        showMessage("Post Created! ID: " + data.id);
    } else {
        showMessage("Error: " + data.error, true);
    }
}
```

- Get Post



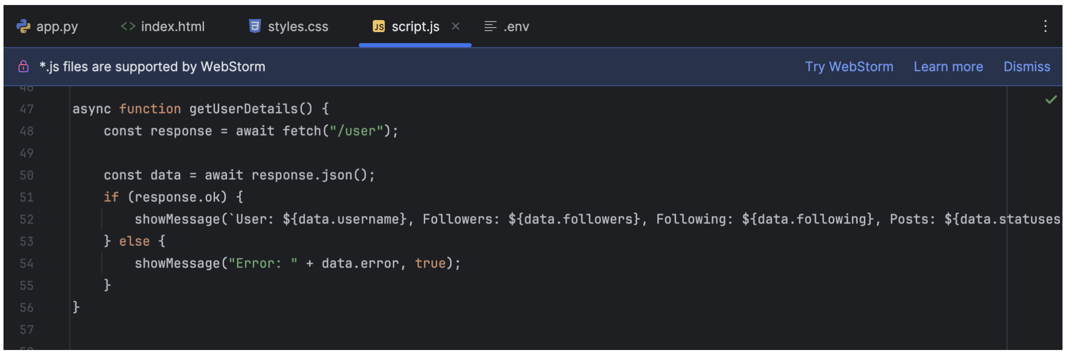
```
async function getPost() {
    const postId = document.getElementById("postId").value;
    const response = await fetch(`/get/${postId}`);

    const data = await response.json();
    if (response.ok) {
        showMessage("Post: " + data.content + " (Created At: " + data.created_at + ")");
    } else {
        showMessage("Error: " + data.error, true);
    }
}
```

- Delete Post



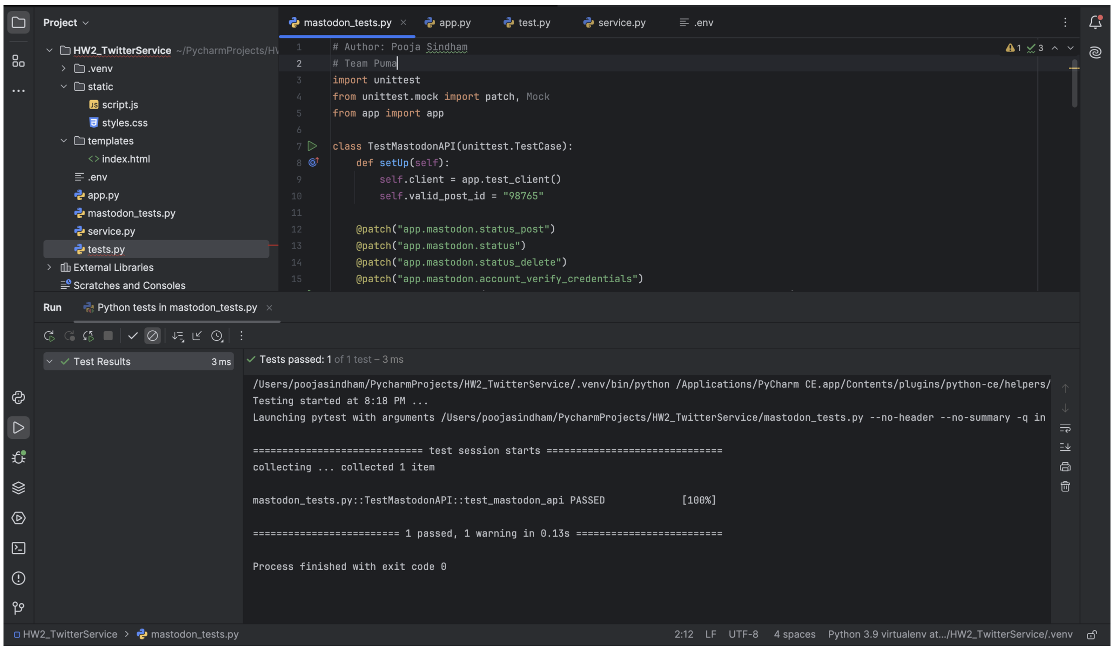
```

async function deletePost() {
    const postId = document.getElementById("postId").value;
    const response = await fetch(`/delete/${postId}`, { method: "DELETE" });

    const data = await response.json();
    if (response.ok) {
        showMessage("Post Deleted!");
    } else {
        showMessage("Error: " + data.error, true);
    }
}
```

- Get User Details



```
async function getUserDetails() {
    const response = await fetch("/user");

    const data = await response.json();
    if (response.ok) {
        showMessage(`User: ${data.username}, Followers: ${data.followers}, Following: ${data.following}, Posts: ${data.statuses
    } else {
        showMessage("Error: " + data.error, true);
    }
}
```

D. Unit tests.



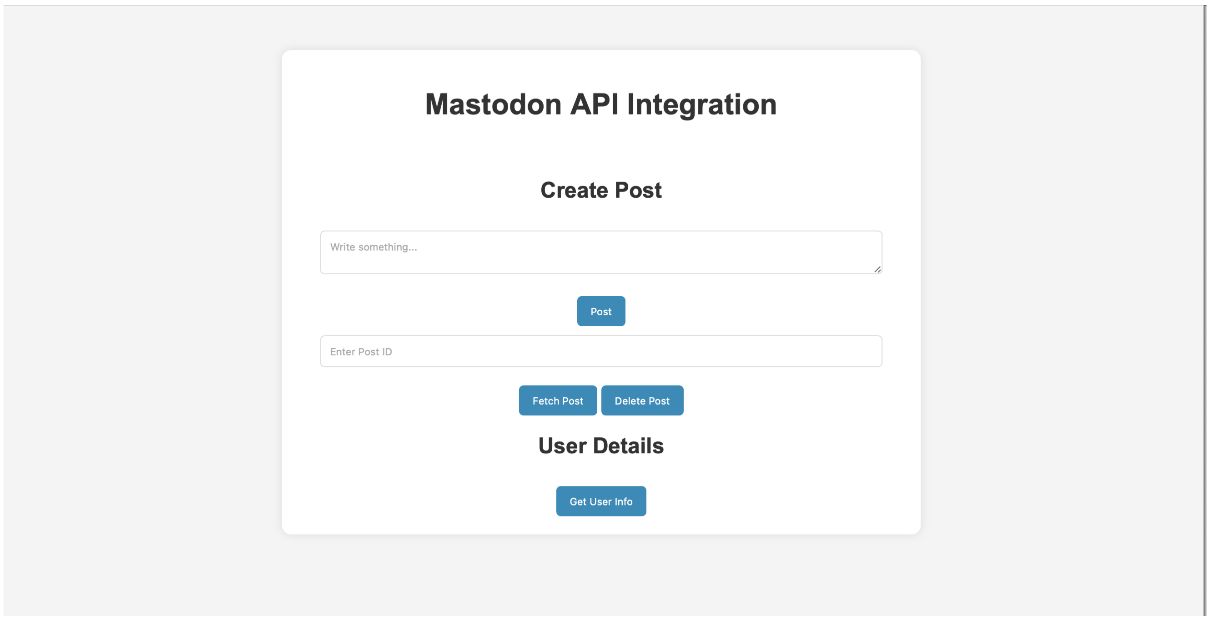
```python
# Author: Pooja Sindham
# Team Puma
import unittest
from unittest.mock import patch, Mock
from app import app

class TestMastodonAPI(unittest.TestCase):
    def setUp(self):
        self.client = app.test_client()
        self.valid_post_id = "98765"

    @patch("app.mastodon.status_post")
    @patch("app.mastodon.status")
    @patch("app.mastodon.status_delete")
    @patch("app.mastodon.account_verify_credentials")
```

Tests passed: 1 of 1 test – 3 ms

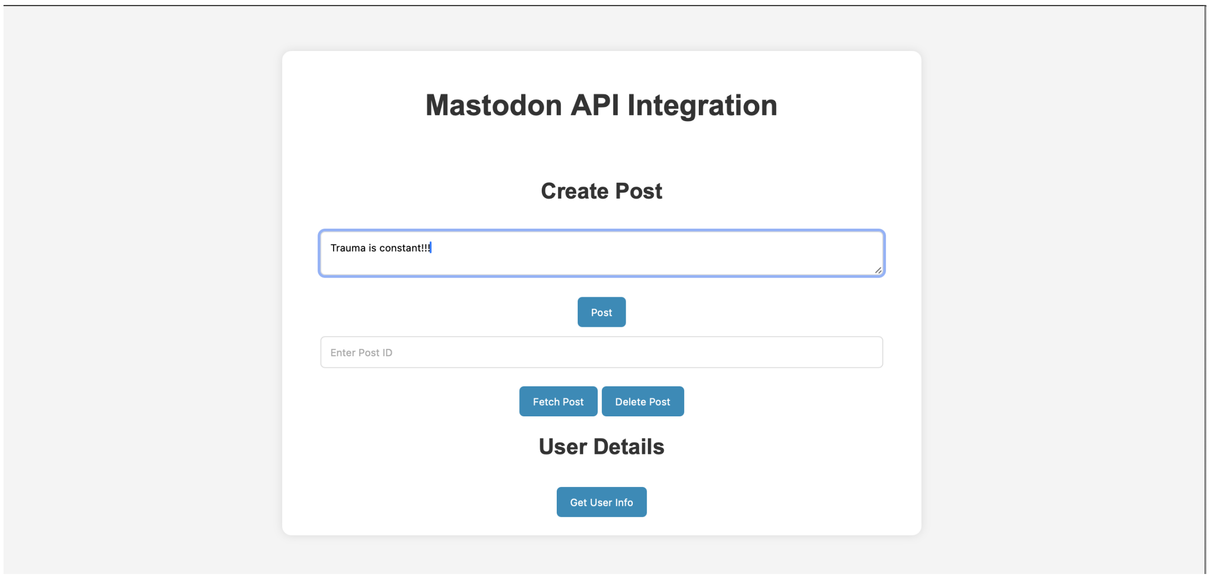
```
/Users/poojasindham/PycharmProjects/HW2_TwitterService/.venv/bin/python /Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/
Testing started at 8:18 PM ...
Launching pytest with arguments /Users/poojasindham/PycharmProjects/HW2_TwitterService/mastodon_tests.py --no-header --no-summary -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)

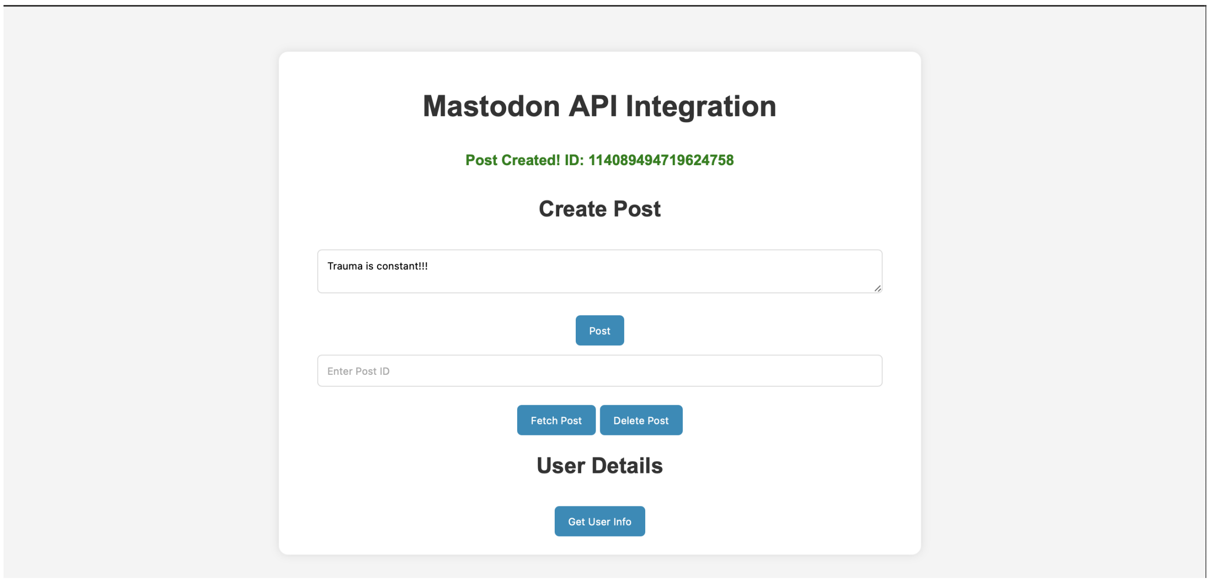
1. Home Page



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



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



Mastodon API Integration

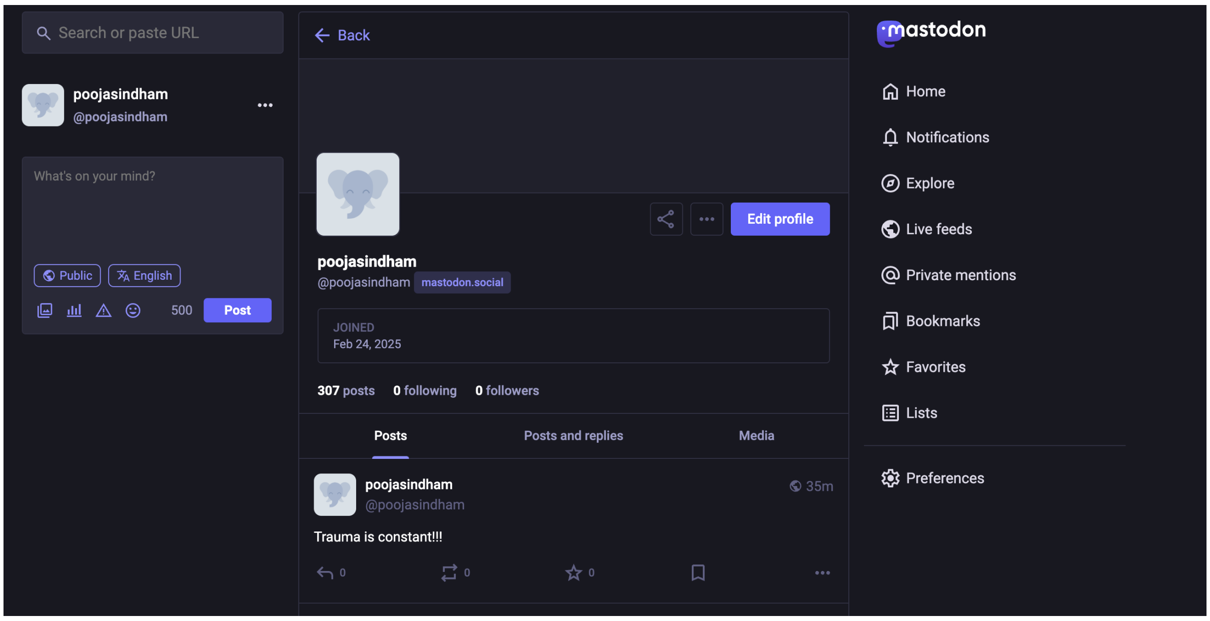
Post Created! ID: 114089494719624758

Create Post

Trauma is constant!!!

User Details

You can see it posted on mastodon social.



poojasindham
@poojasindham mastodon.social

JOINED
Feb 24, 2025

307 posts 0 following 0 followers

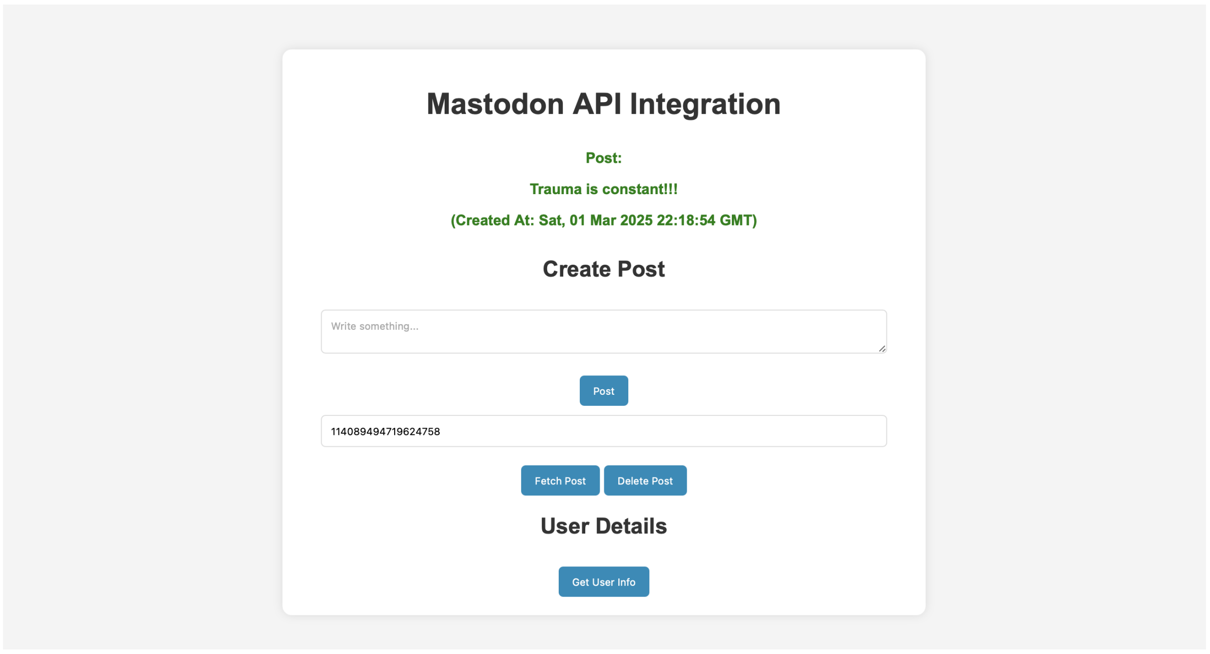
poojasindham
@poojasindham 35m

Trauma is constant!!!

0 0 0

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)

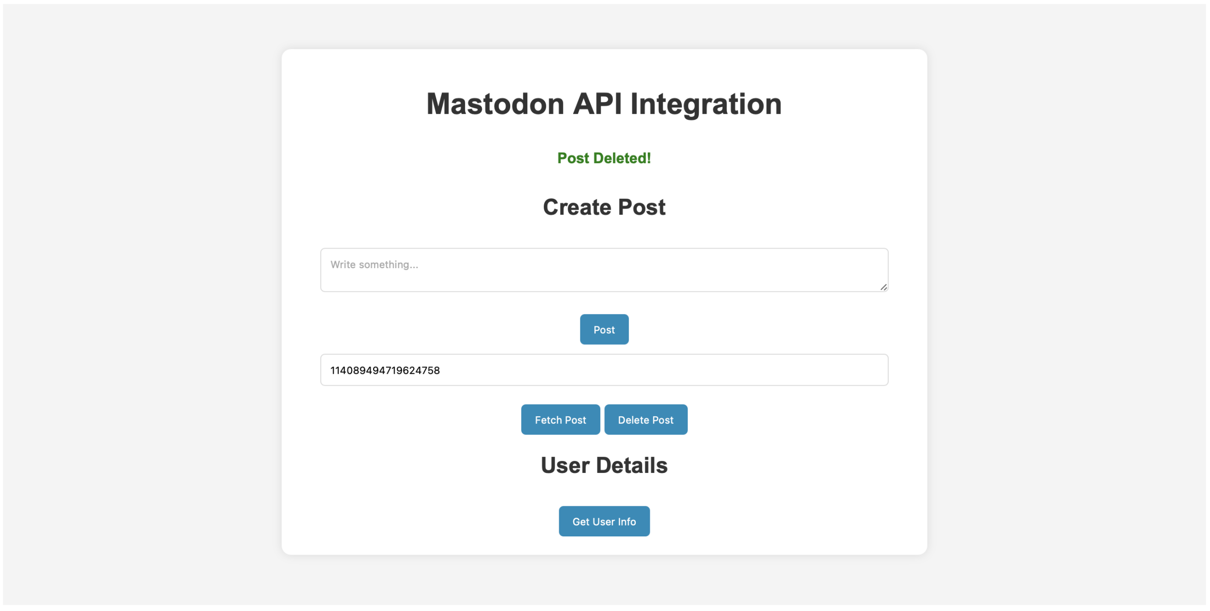
Create Post

Write something...

114089494719624758

User Details

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



Mastodon API Integration

Post Deleted!

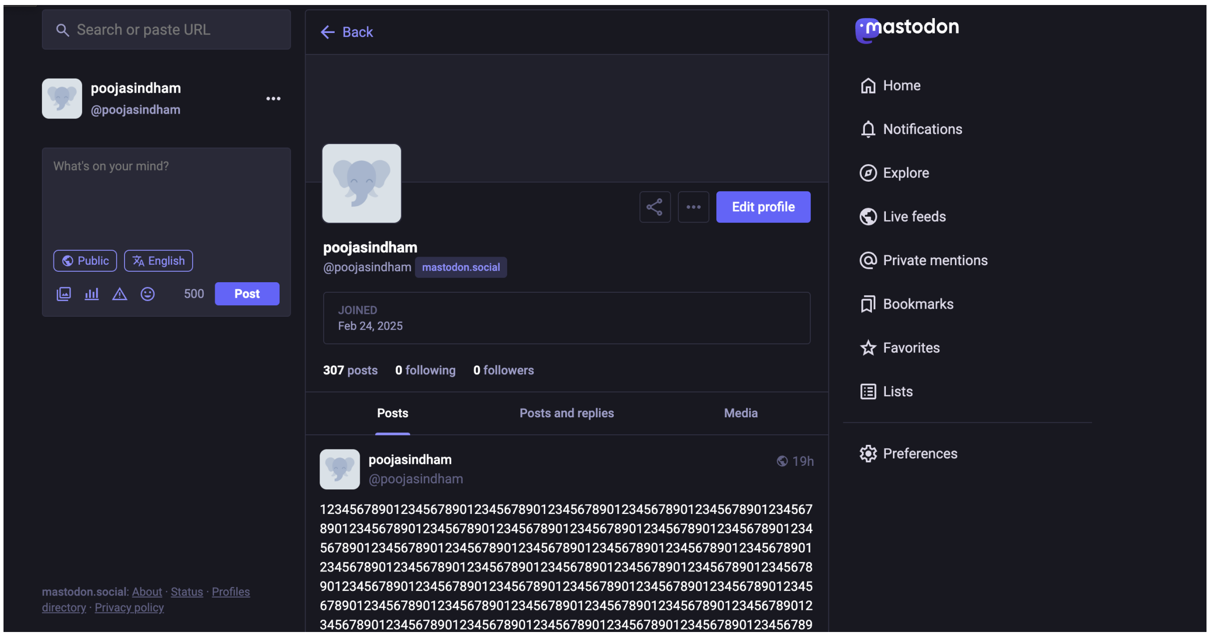
Create Post

Write something...

114089494719624758

User Details

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



poojasindham
@poojasindham mastodon.social

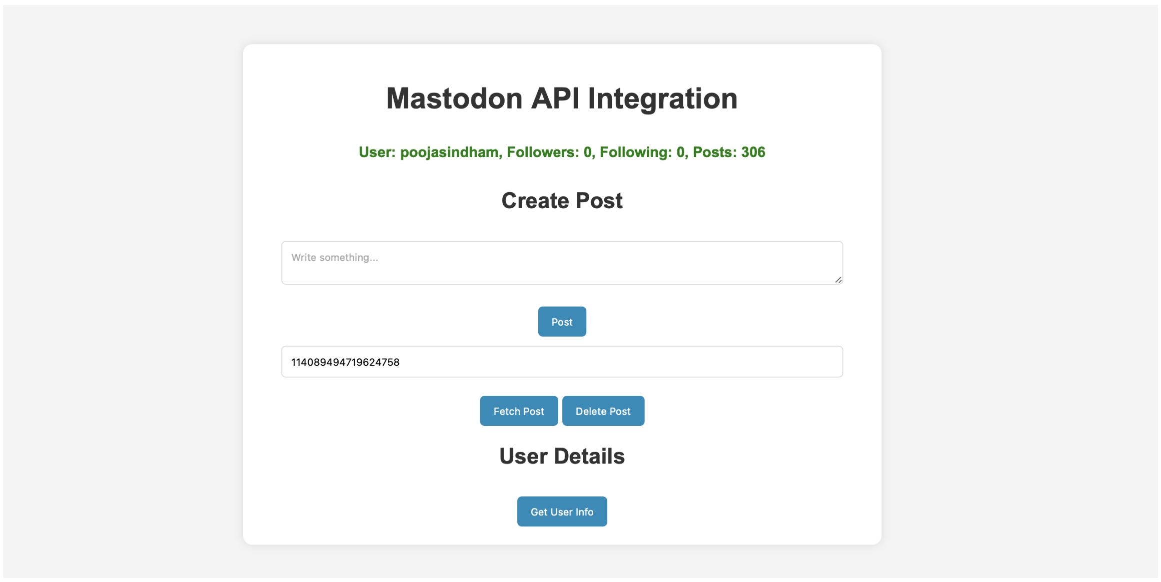
JOINED
Feb 24, 2025

307 posts 0 following 0 followers

poojasindham
@poojasindham 19h

12345678901234567890123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234567890123456789012345678901234567890123456789

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



Mastodon API Integration

User: poojasindham, Followers: 0, Following: 0, Posts: 306

Create Post

1140894947196247 58

User Details