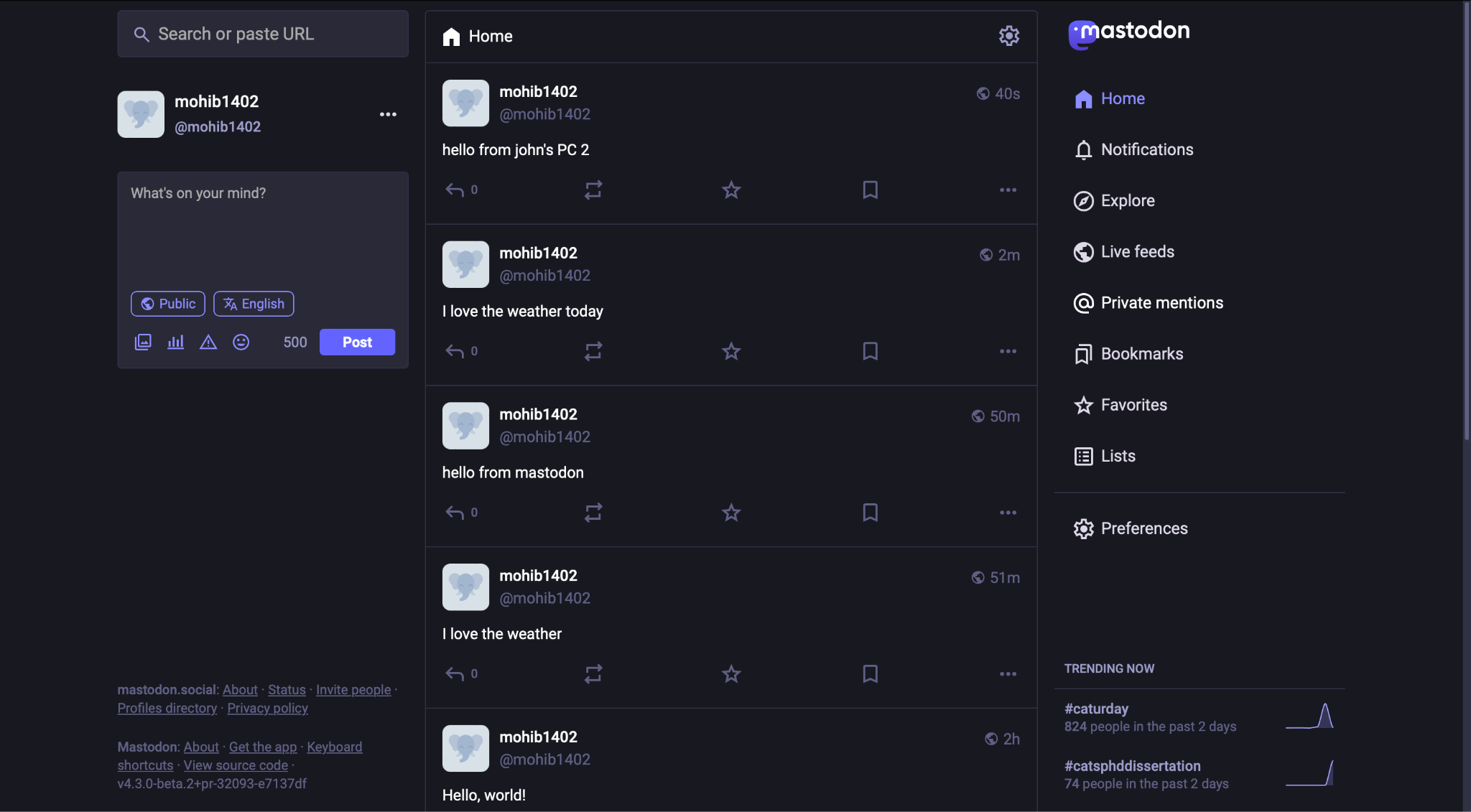
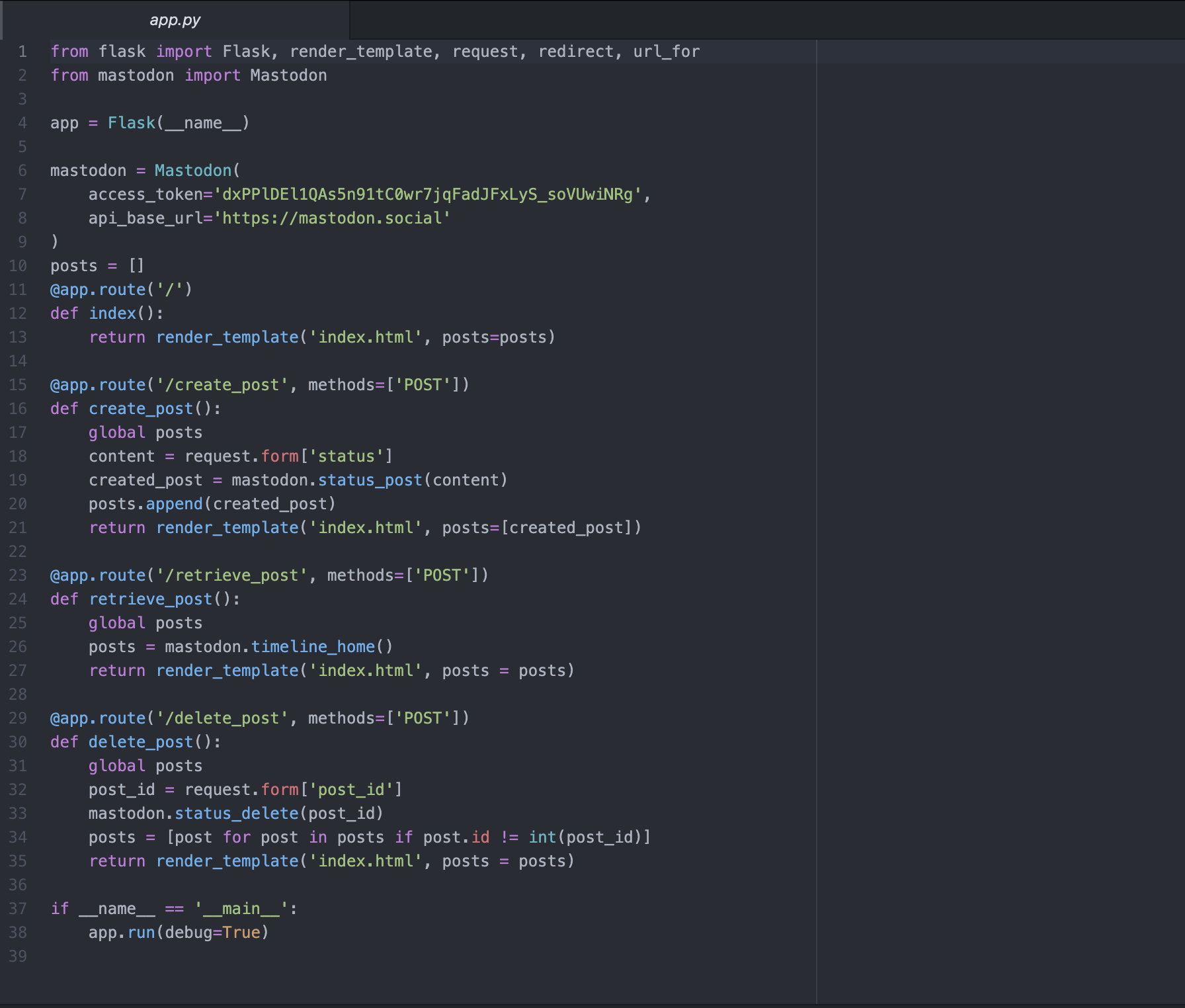
**Submitted by Team Alpha (Data Science Cohort)  
Mohibkhan Pathan and Praful John   
Link to the github repository:** [**https://github.com/Mohib1402/CMPE272HW2TwitterService.git**](https://github.com/Mohib1402/CMPE272HW2TwitterService.git) **Work done by Mohibkhan Pathan (I initially had it as a Twitter Service, so that is how it’s been shown but Praful made the name change to Mastodon in the later commits):**

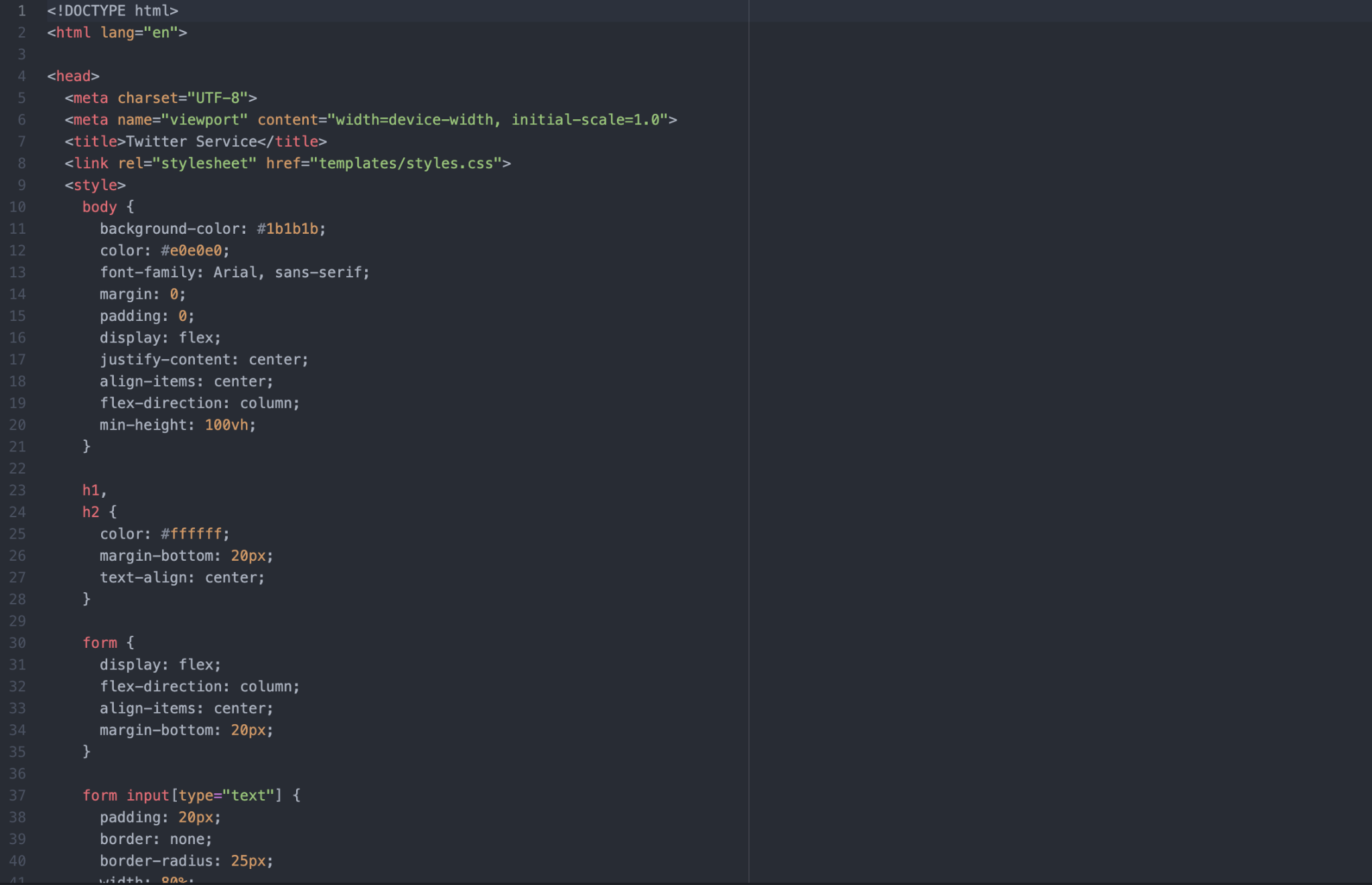
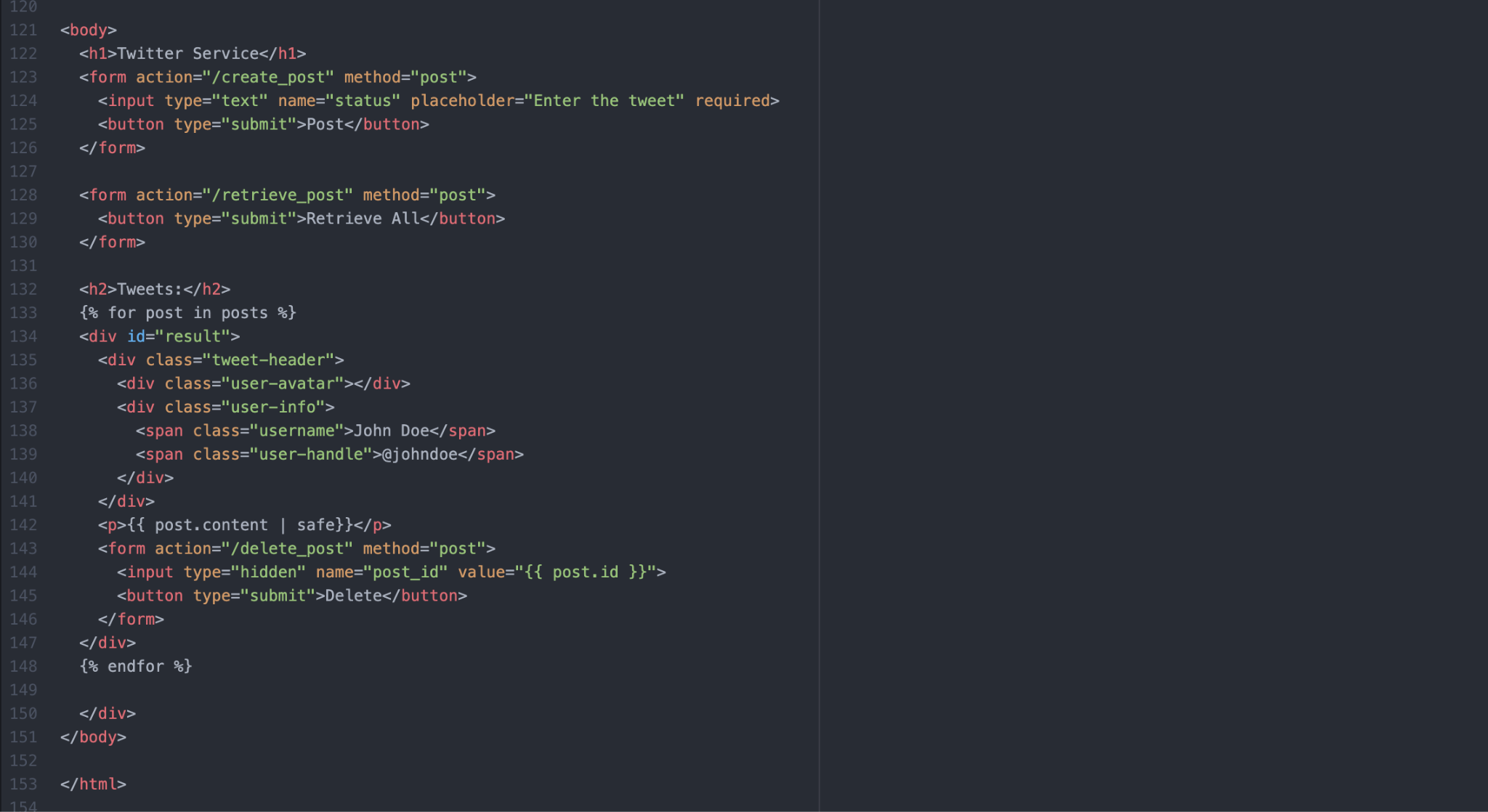
1. **Mastodon Setup (mastodon.social):**

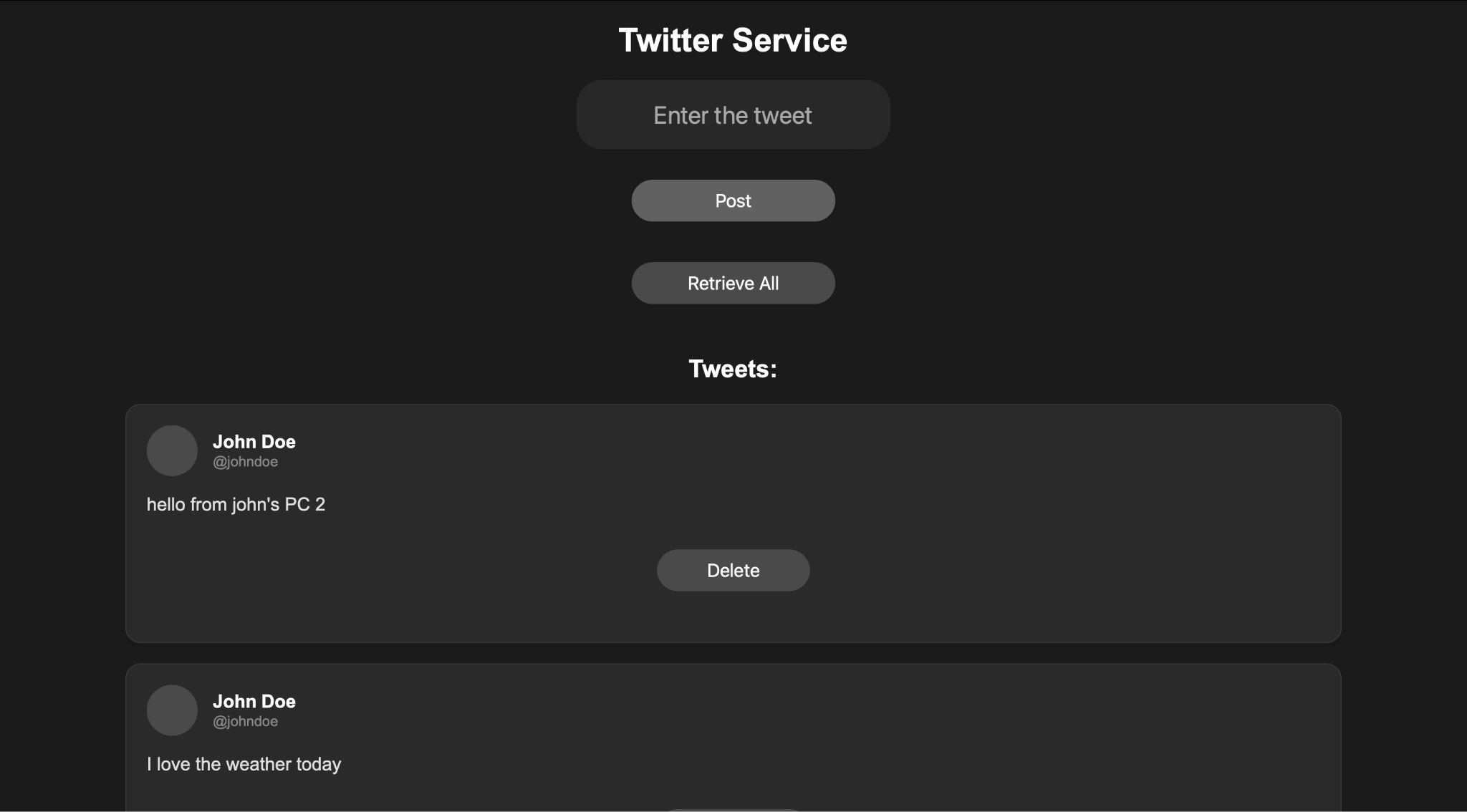
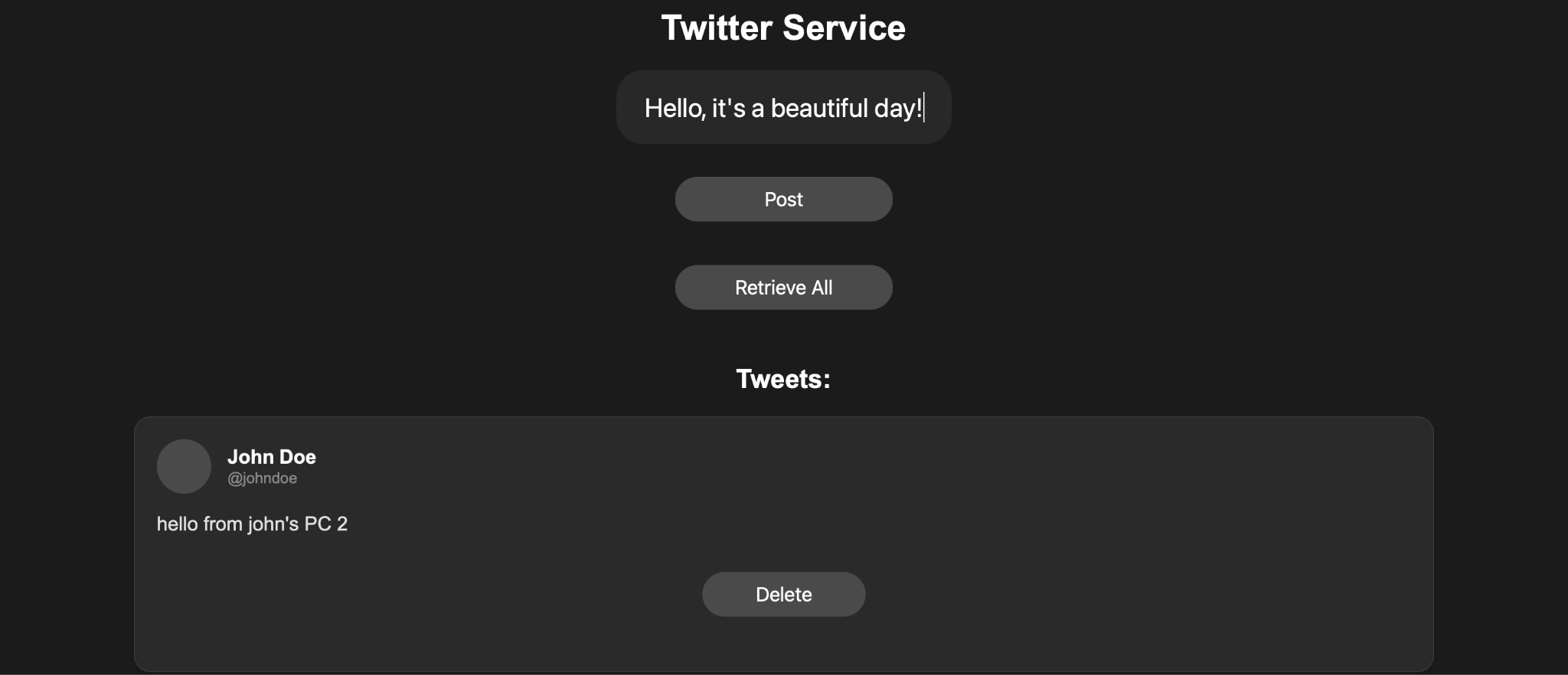
****

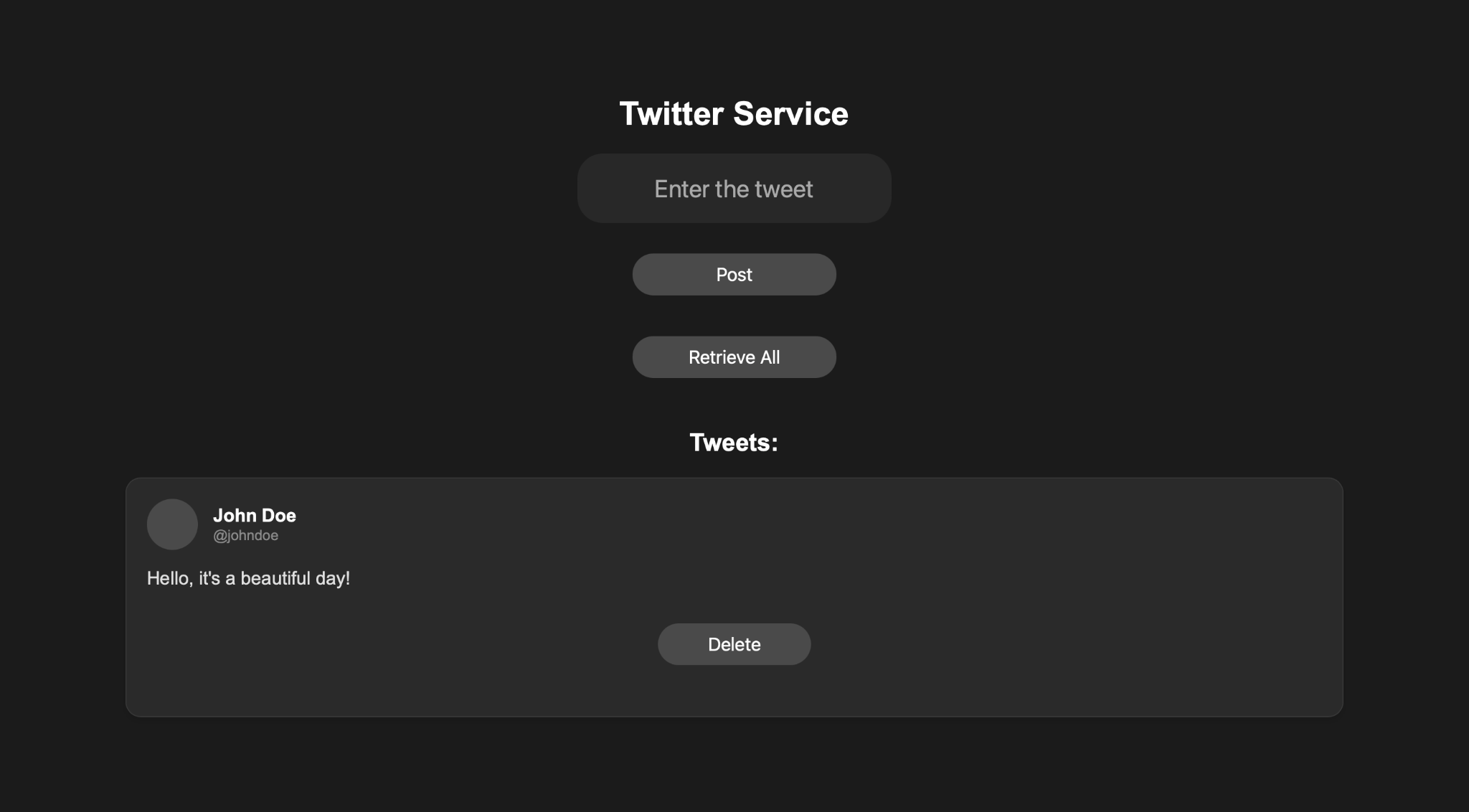
1. **App backend setup and relevant methods for creating a post, retrieving all the posts and deleting the posts, this was further edited by Praful to add a method to retrieve a post by its postId on Mastodon:**



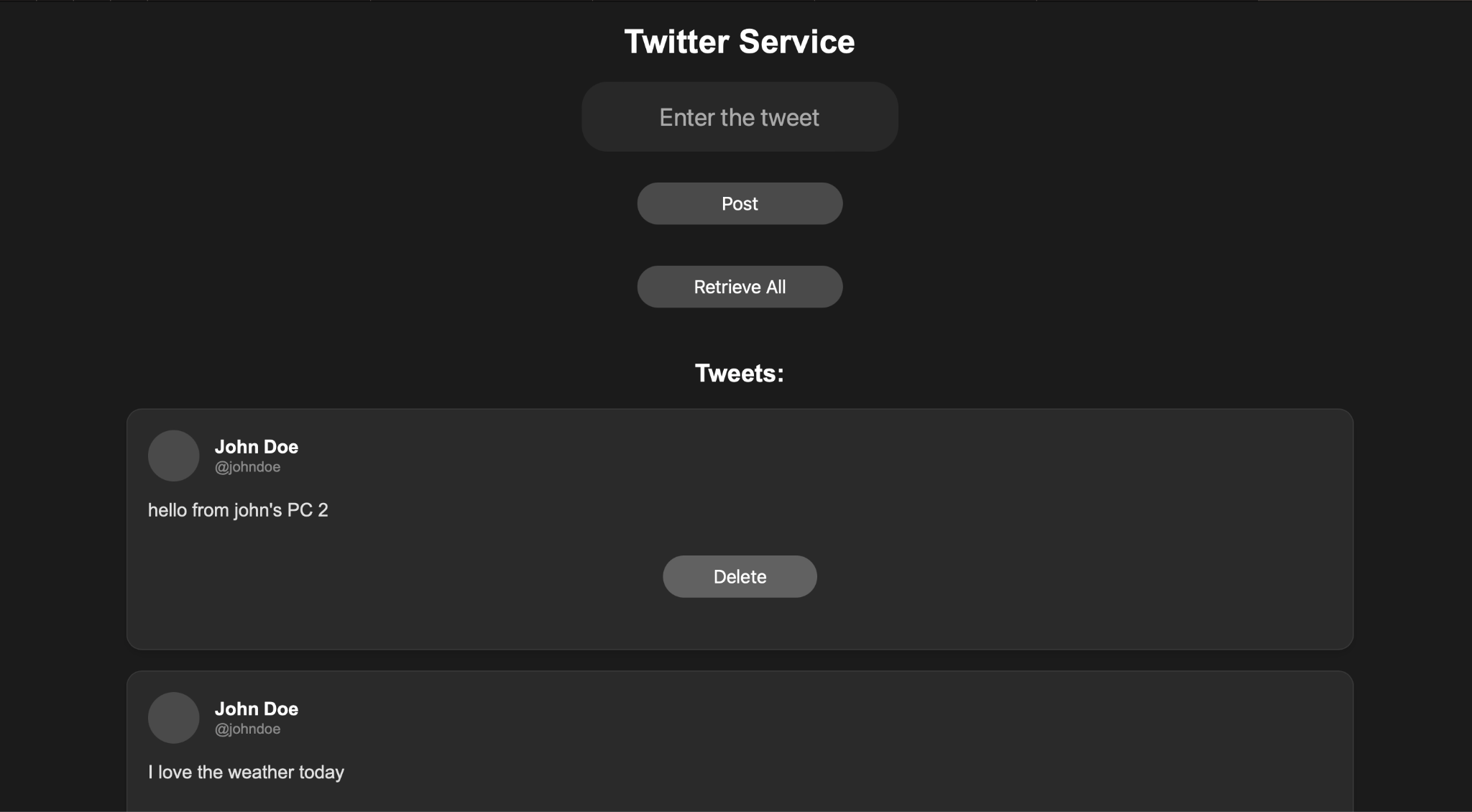
1. index method is the routing for the home page, it renders the index.html page where the user can create a post, delete a post and retrieve a post by its id or retrieve all posts.
2. create\_post method works in a way that when the user submits a post on the frontend side of the web-app, the user takes the text as a parameter and passes on to this create\_post method along with the route. Then the method uses mastodon.py’s status\_post method to create a post. And the post is further added to the list of already existing posts in Mastadon’s API.
3. retrieve\_post basically checks the Mastodon API’s timeline to see what are all the posts available and to do that, it uses timeline\_home method which returns a dictionary of posts which is then rendered in the index.html template.
4. delete\_post method works in a similar way as the create\_post as it is given the post\_id along with the route from the delete button, it uses the mastodon.py’s status\_delete method to delete any post that has been created, this would further assign a non integer value to the post\_id. Finally, we render the index.html file with all the posts that have an integer post\_id.
5. **Setting up index.html with inline CSS and fulfilling the UI related tasks:**

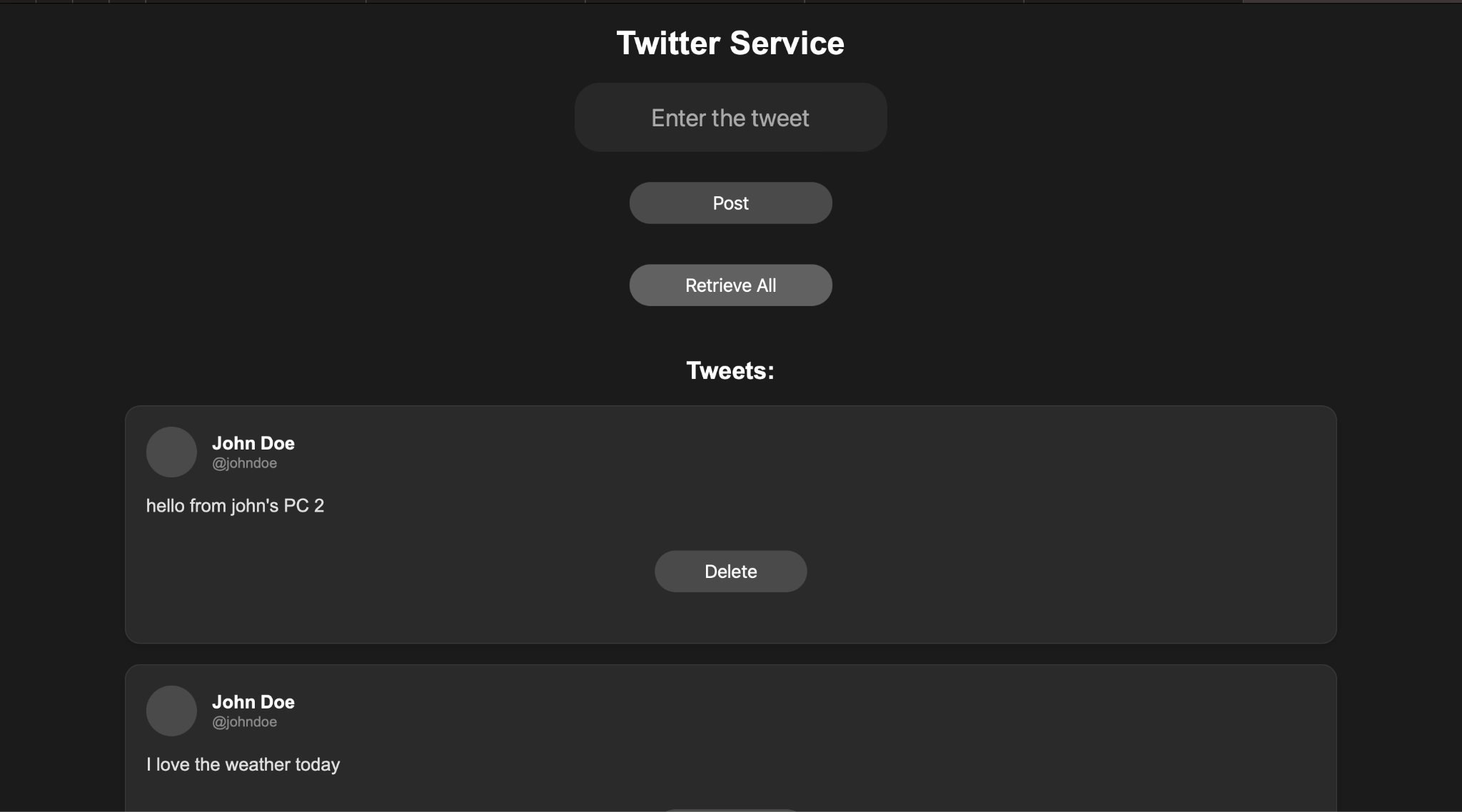
****

1. **App UI that performs the necessary tasks and this includes an addition from Praful which will be shown in his section in this doc:**
2. **Posting a Tweet**

****

1. **Deleting the tweets**

****

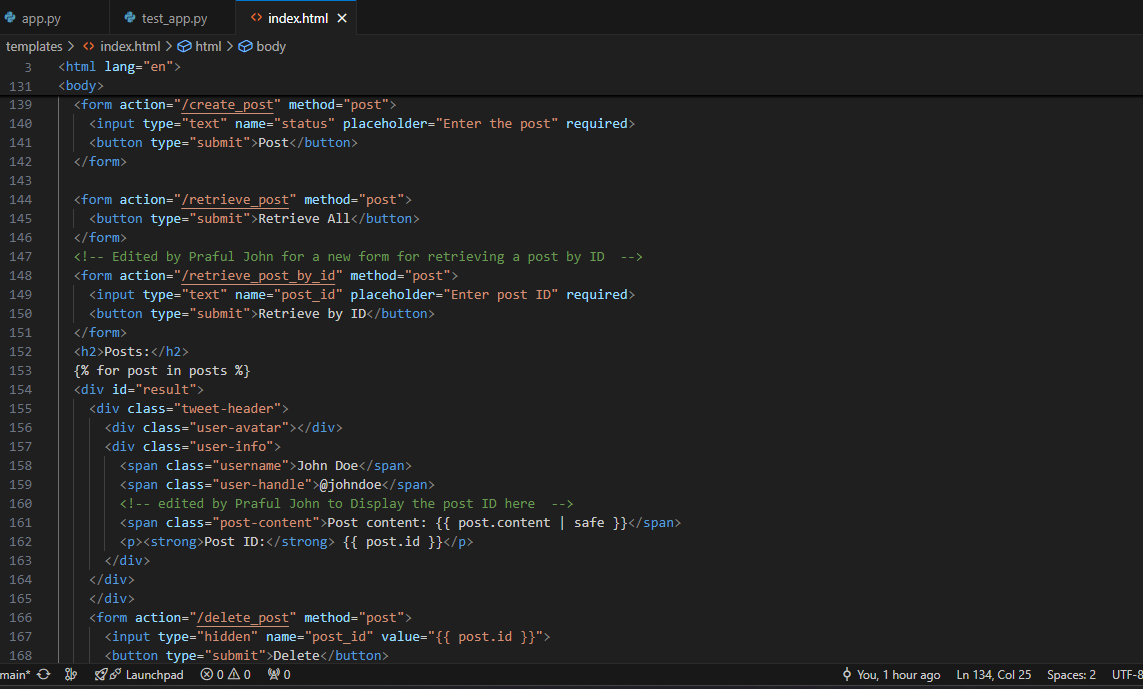
1. **Retrieving the tweets**

**Work done by Praful John**

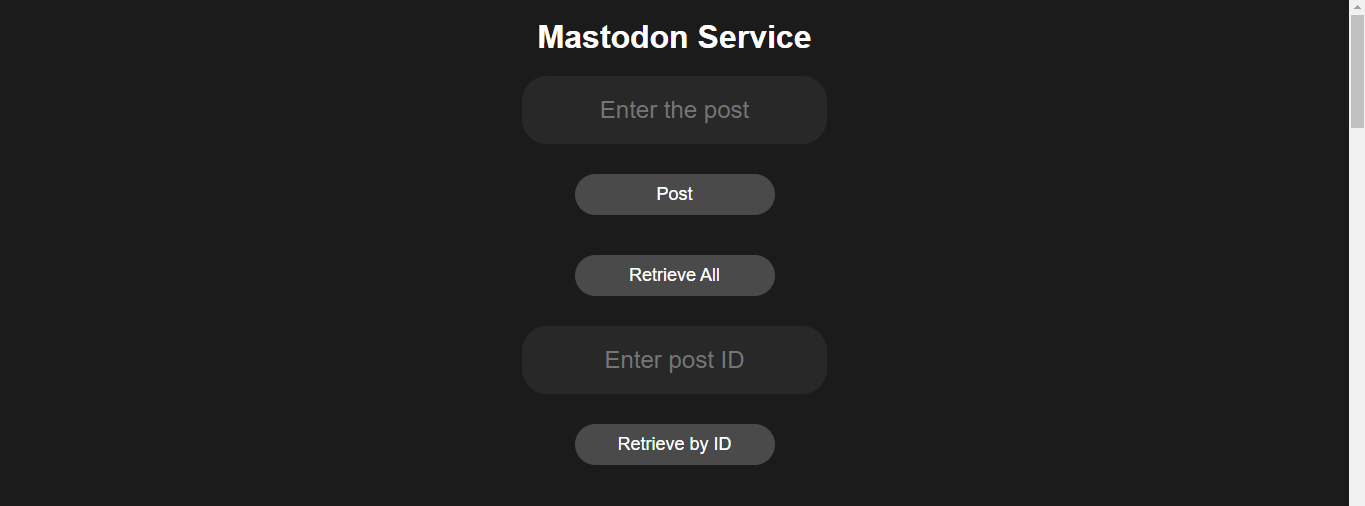
**Table of contents:**

1. **Modified the UI to show the Post ID of a Post**
2. **Added the functionality of retrieval by Post ID.**
3. **Handled the validations of the post creation and cases like API rate limits or invalid inputs.**
4. **Created the Unit tests that demonstrate the correctness of our API interactions using unittest.**

**1) Modified the UI to show the Post ID of a Post**

****

**Fig: The code that modifies the UI for the new button**

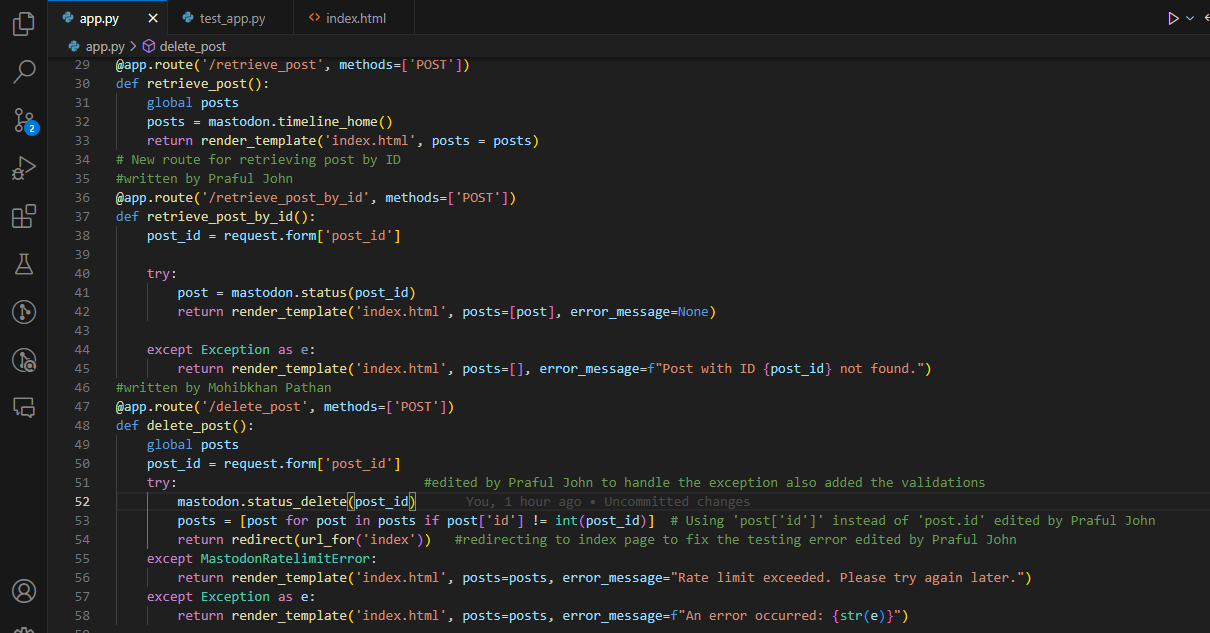
****

**Fig: after the changes this is how the UI looks like**

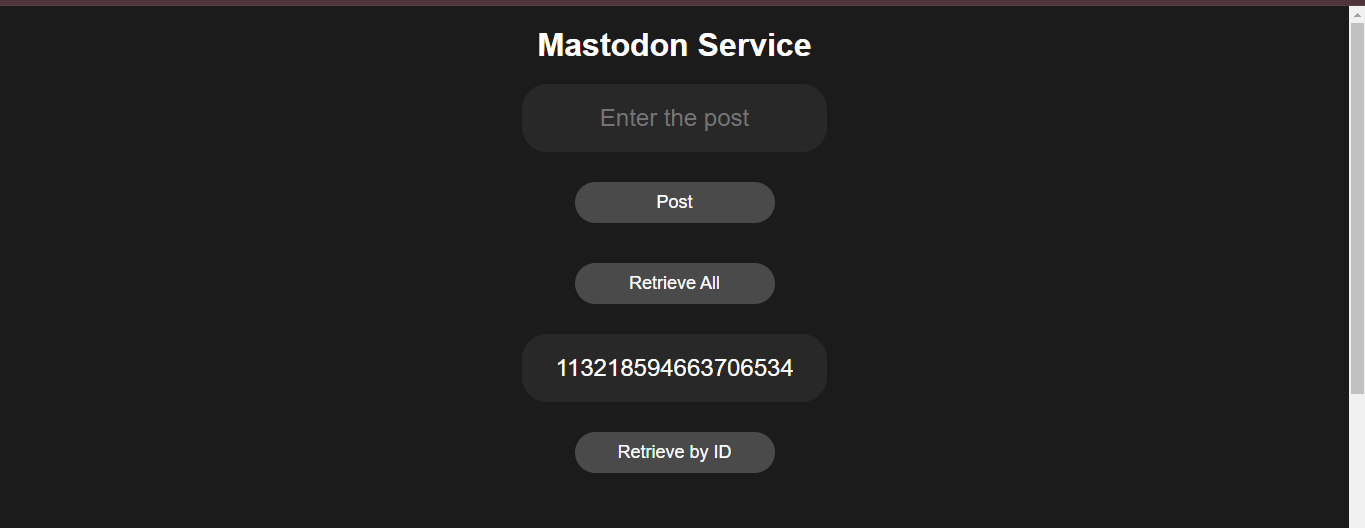
**2) Added the functionality of retrieval by Post ID.**

**About the function retrieve\_post\_by\_id**

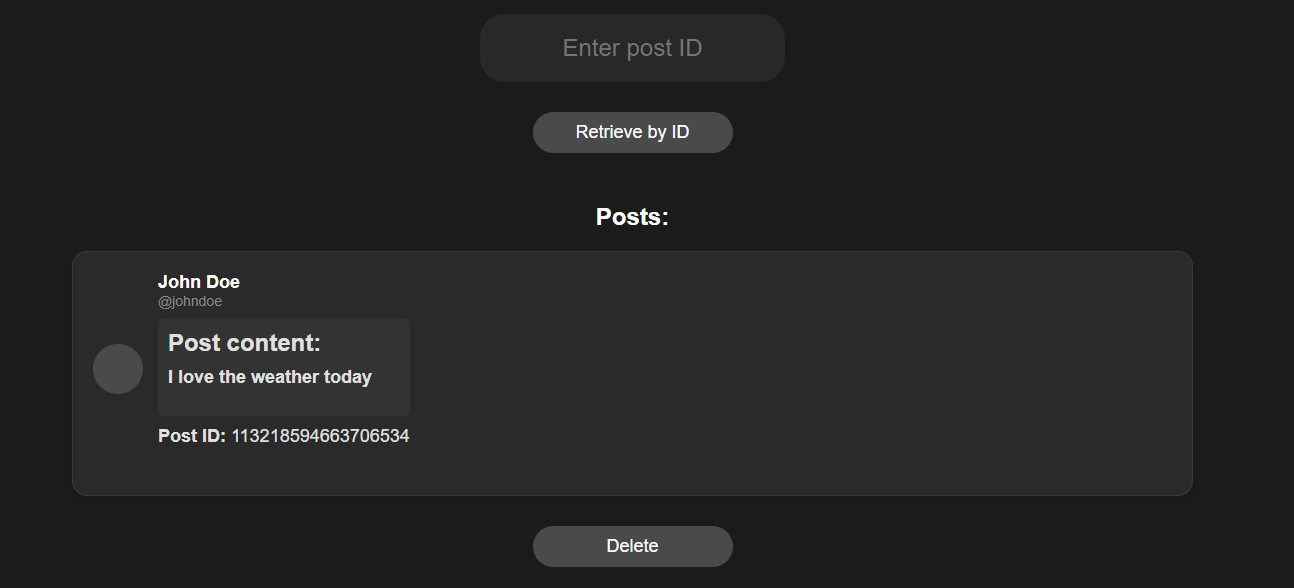
* Extracts the post\_id from the form, fetches the post using mastodon.status(post\_id), and renders the result.
* If the post is found, it displays it on the index.html page with the post. If the post is not found or an error occurs, it shows an error message indicating that the post with the given ID was not found.

****

**Fig: the code to the retrieve post by id**

****

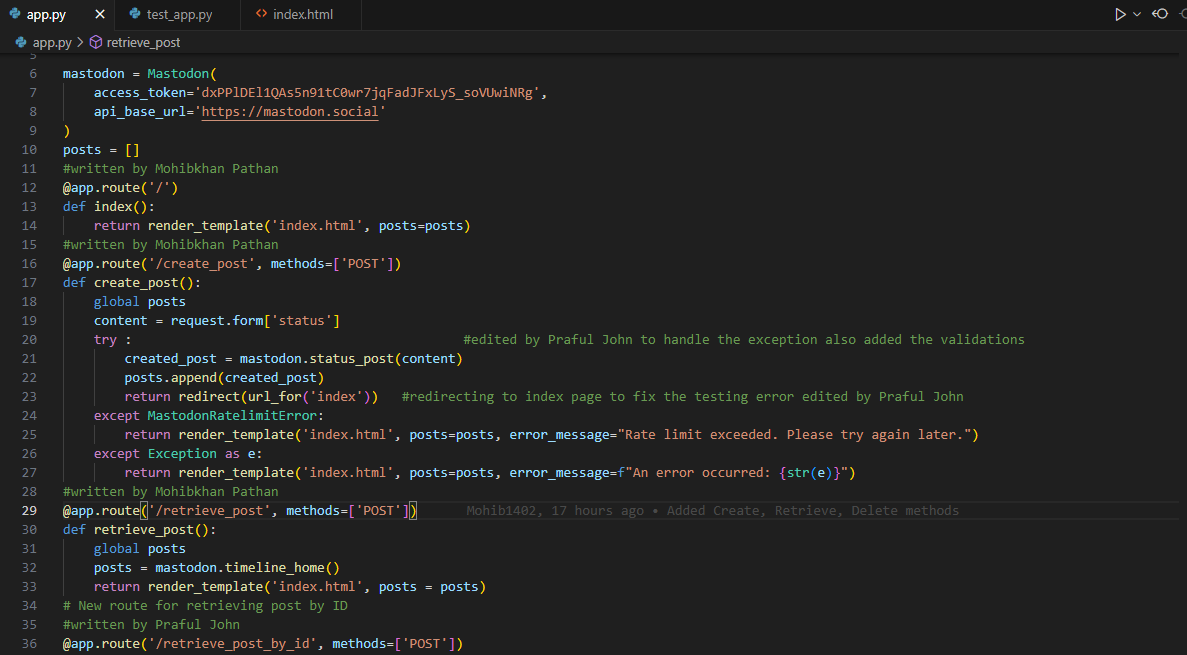
**Fig: entering the id in the new functionality**

****

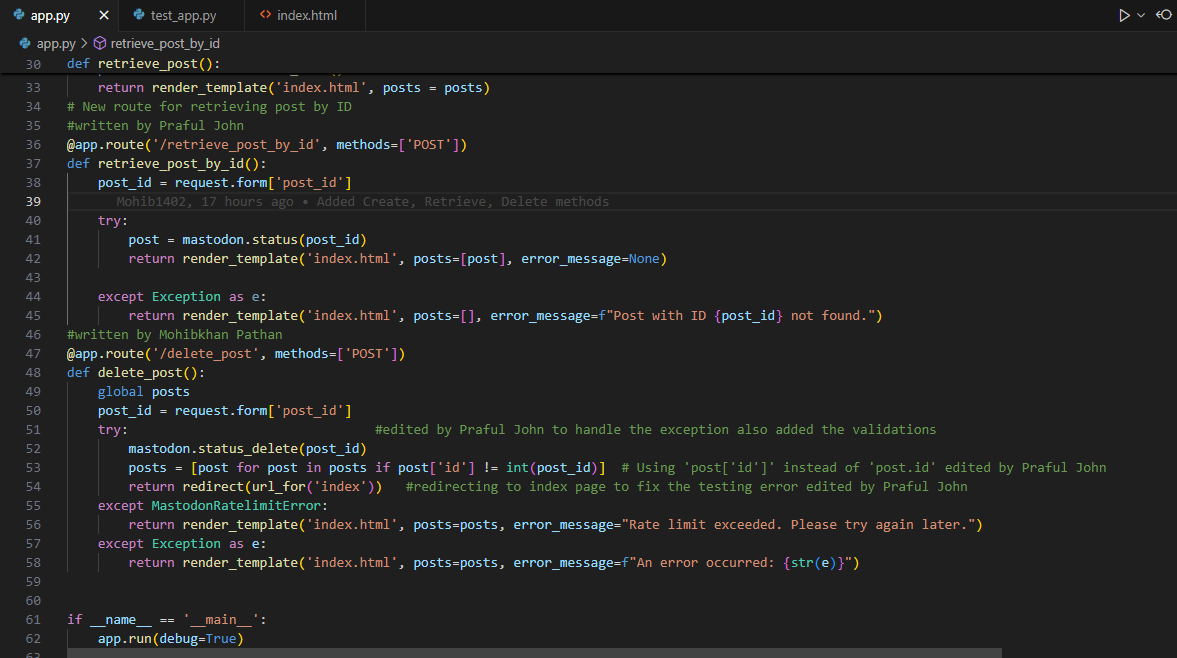
**FIg: the post with that id is retrieved.**

**3) Handled the validations of the post creation and cases like API rate limits or invalid inputs.**

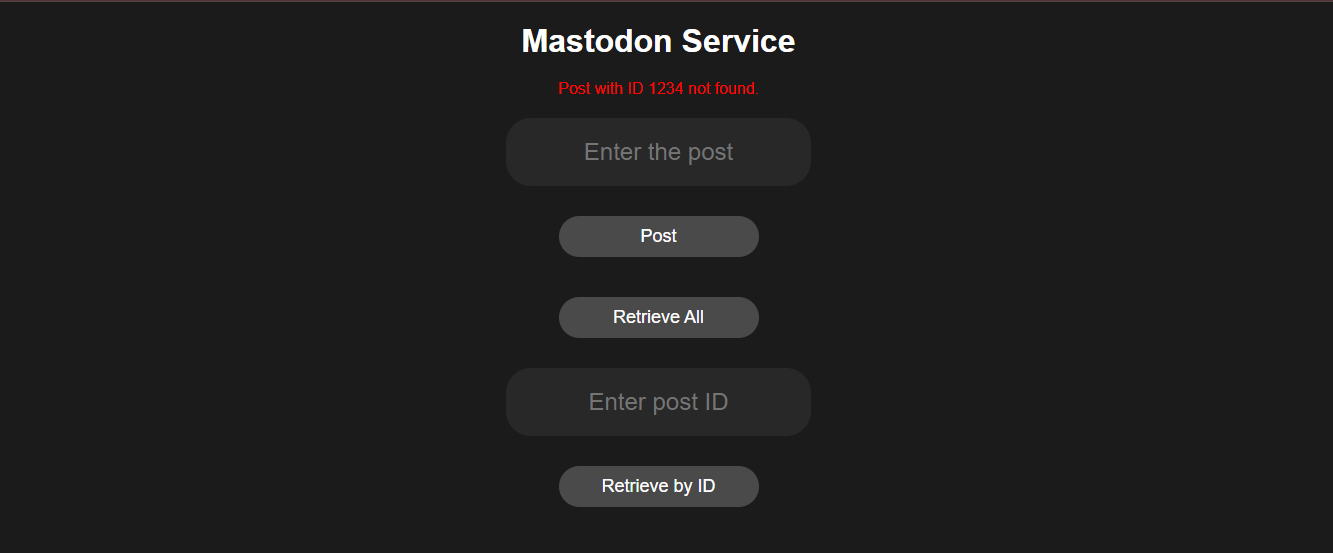
* **Handling the Rate time limit error**

****

* **Handled the wrong inputs and rendered the index.html page to fix the error while testing**

****

**Fig: Code for Handled the wrong inputs and rendered the index.html page to fix the error while testing**

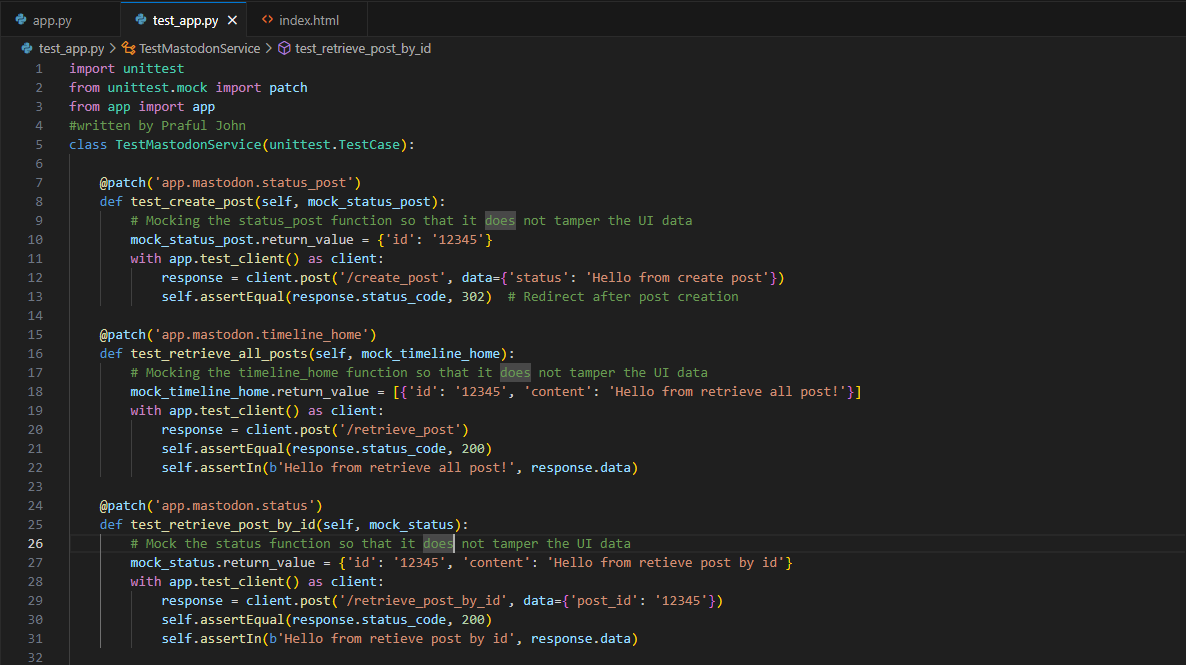
****

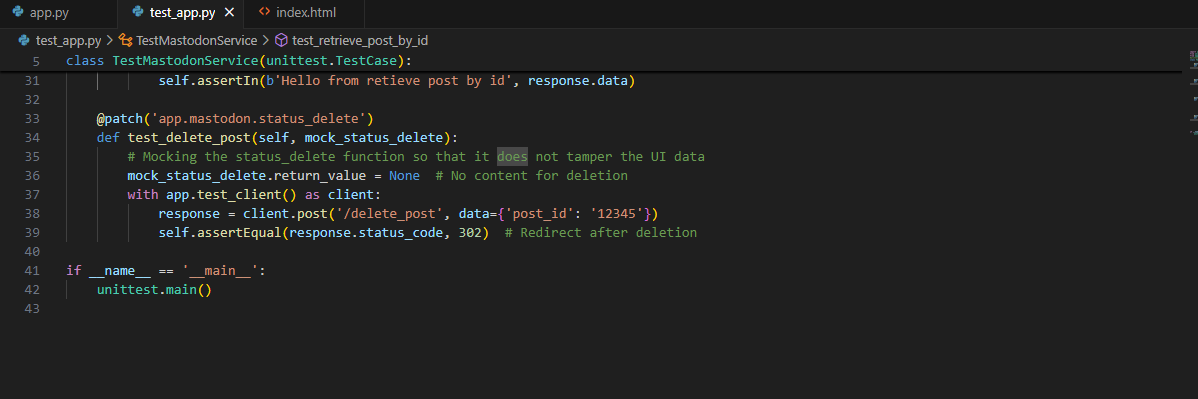
**Fig: UI displaying the message that the ID doesn’t exist.**

**4) Developed the Unit tests that demonstrate the correctness of our API interactions using unittest.**

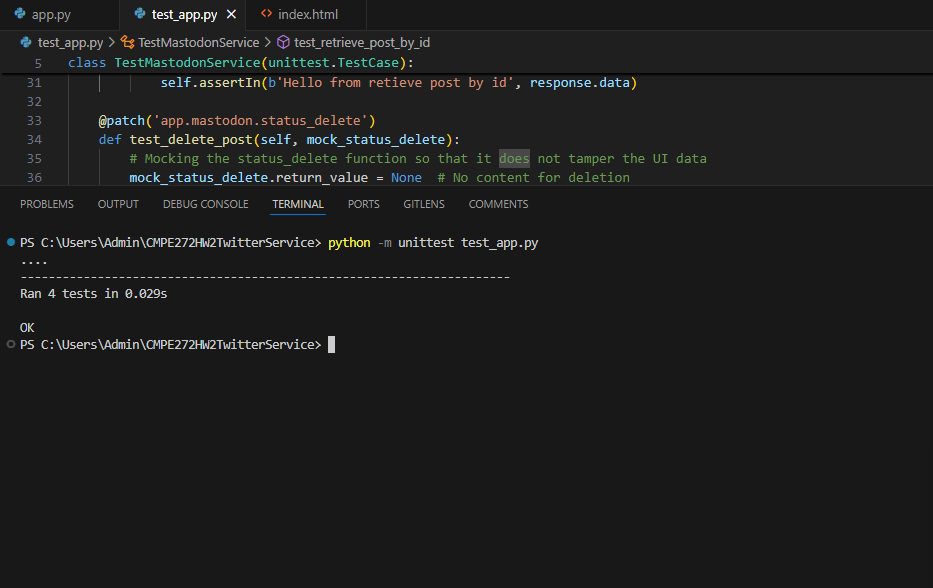
**About the test\_app.py file**

* **test\_create\_post**:Mocks mastodon.status\_post() to prevent real API calls. It sends data to the /create\_post API and verifies that the response redirects .
* **test\_retrieve\_all\_posts**:Mocks mastodon.timeline\_home() to simulate retrieving posts. It tests the /retrieve\_post route and verifies that the mocked post content is returned in the response.
* **test\_retrieve\_post\_by\_id**:Mocks mastodon.status() to simulate fetching a post by ID. It posts an ID to the /retrieve\_post\_by\_id route and checks if the correct post content is present in the response.
* **test\_delete\_pos**t:Mocks mastodon.status\_delete() to simulate deleting a post. It sends a post ID to the /delete\_post route and verifies that the response redirects after deletion.

****

****

**Figs: the test\_app file**

**Results: All test cases passed**