

Homework #2 – Twitter Service

Group:

Parth Patel

Shailen Sutradhar

Gautam Santhanu Thampy

Dev Patel

Class: CMPE272 Sec 03

Prof: Andrew Bond

Date: Feb 24th, 2025

Source code

The source code for this assignment is stored on GitHub.

GitHub: <https://github.com/parthpatelsjsu/CMPE272-Mastodon>

Unit Tests

Testing library: Vitest

Test cases: mastodon/src/App.test.tsx.

Created by: Parth Patel

There are several functions implemented in this assignment. There is one or more test cases created for each function to test the functionality. These test cases are stored at the path provided above.

Vitest was chosen because it is a lightweight testing framework built specifically for projects using Vite, which was used to create and run this project.

There are a total of 7 test cases. They are as follows:

1. Renders the app title
2. Shows error when trying to create an empty post
3. Shows error when trying to fetch a post without an ID
4. Shows error when trying to delete a post without an ID
5. Fetches post using GET operation when the fetch button is clicked
6. Performs the create post operation when the create button is clicked
7. Performs the delete operation when the delete button is clicked

The test cases are run using the CLI command “npx vitest” which runs the test cases as shown below

```
✓ src/App.test.tsx (7 tests) 78ms
  ✓ App Component > renders the app title
  ✓ App Component > shows error when trying to create an empty post
  ✓ App Component > shows error when trying to fetch a post without an ID
  ✓ App Component > shows error when trying to delete a post without an ID
  ✓ App Component > fetches posts using GET operation when the fetch button is clicked
  ✓ App Component > performs the POST operation when the create button is clicked
  ✓ App Component > performs the DELETE operation when confirmed in the modal

Test Files 1 passed (1)
Tests 7 passed (7)
Start at 18:26:30
Duration 556ms (transform 59ms, setup 0ms, collect 143ms, tests 78ms, environment 208ms, prepare 27ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```

Web UI

The assignment has been made public in the GitHub repository and deployed to GitHub pages. It is connected to Mastodon server, so any API request will affect posts on the server.

Hosted on GitHub Pages: <https://parthpatelsjsu.github.io/CMPE272-Mastodon/>

UI Interaction

Creating a post

File: mastodon/src/App.tsx

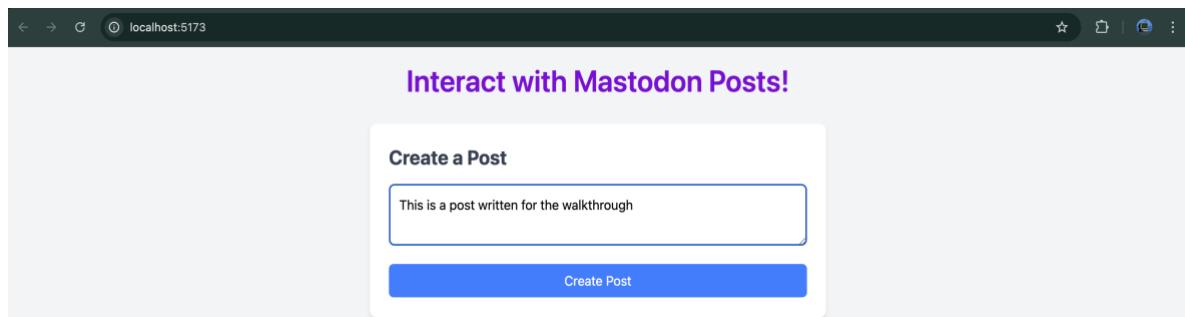
Function: createPost()

Implemented by: Shailen Sutradhar

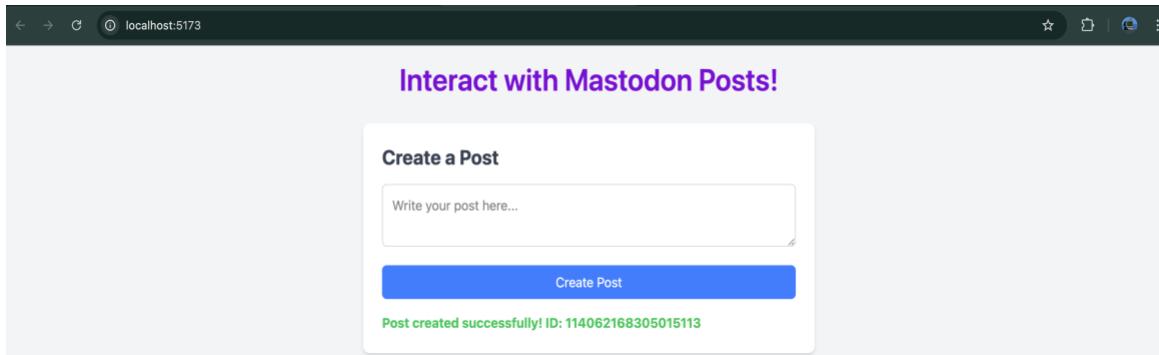
This function takes the contents of the text box as input. If the input contains any non-whitespace characters, a POST request containing the instance, authorization token, and content-type is assembled. The result is a post successfully being created in our Mastodon server, and success is displayed on the UI. If the input is invalid, or if the POST request fails for any reason, the failure along with a reason is displayed on the UI.

The walkthrough is as follows:

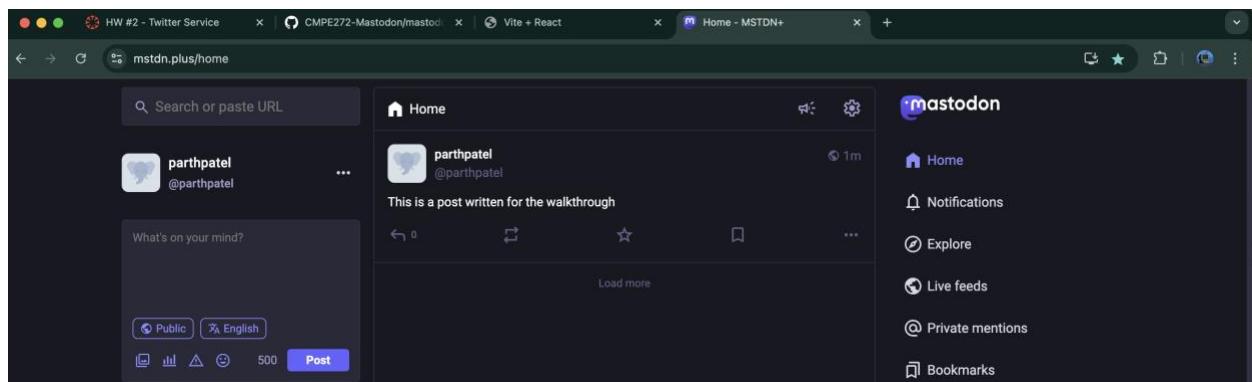
1. Write the post in the textbox



2. Click create (image shows after click)



3. Post shown on <https://mstdn.plus/home>



Fetching a Post

File: mastodon/src/App.tsx

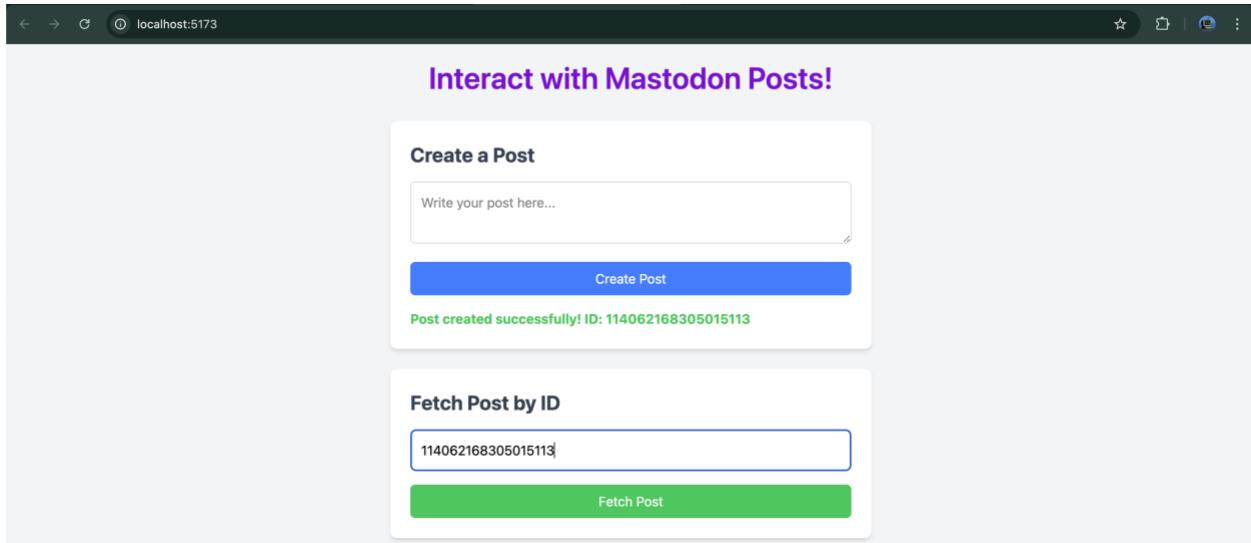
Function: fetchPostById()

Implemented by: Dev Patel

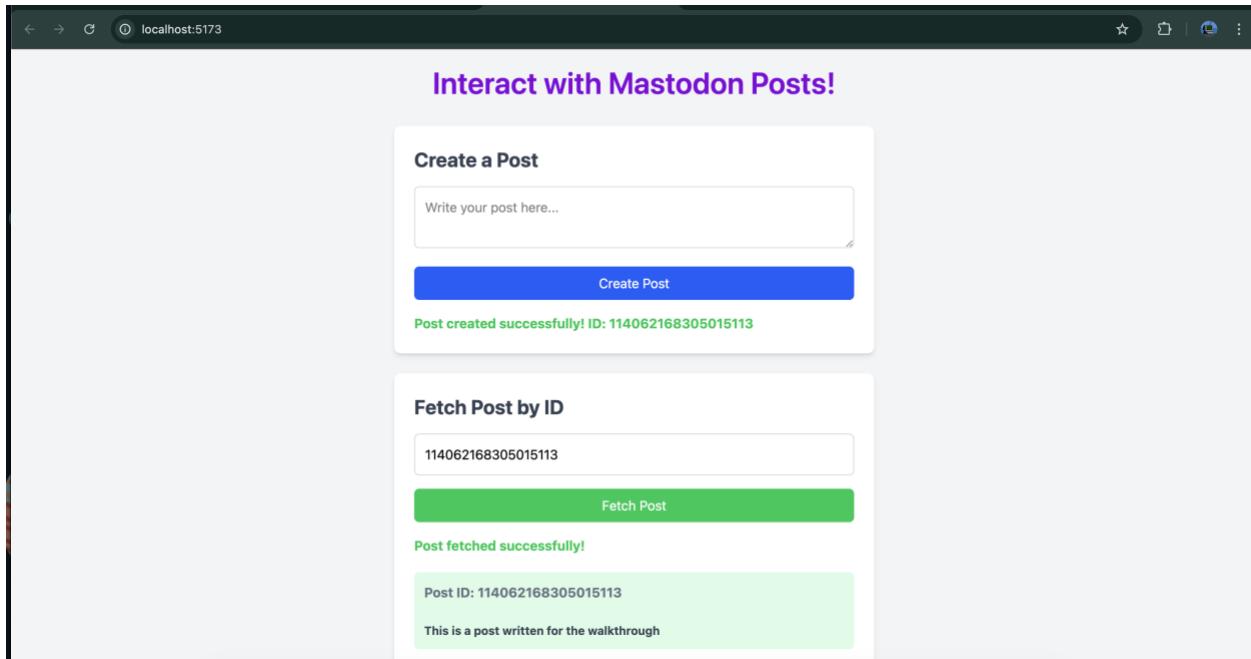
This function takes the contents of fetch ID textbox as input. If the input is a valid ID of a post on our Mastodon server, it assembles a GET request containing the instance, post ID to be fetched, authorization token, and content-type. This request retrieves the post and displays the ID and content on the UI. If the ID is invalid or the GET request fails, a failure message and reason is posted on the UI.

The walkthrough is as follows:

1. Provide an ID of a post in the textbox



2. Click “Fetch Post” button to see the post



Deleting a Post

File: mastodon/src/App.tsx

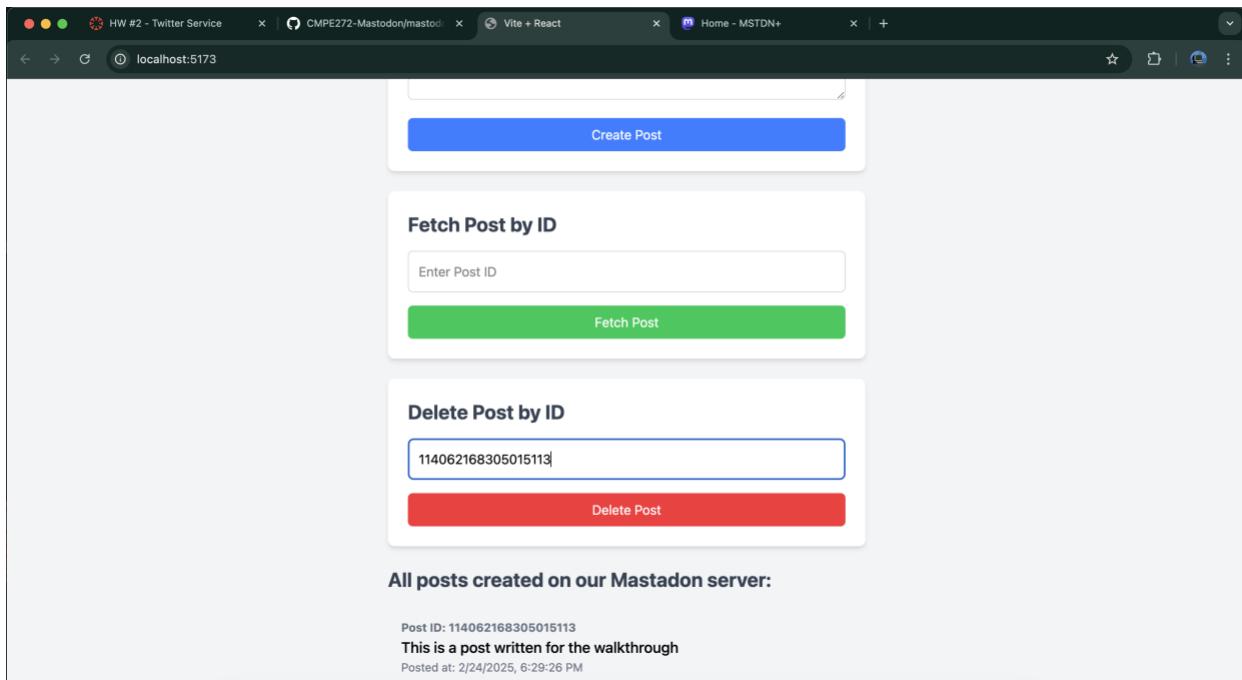
Functions: handleConfirmDelete(), handleDeleteClick()

Implemented by: Gautam Santhanu Thampy

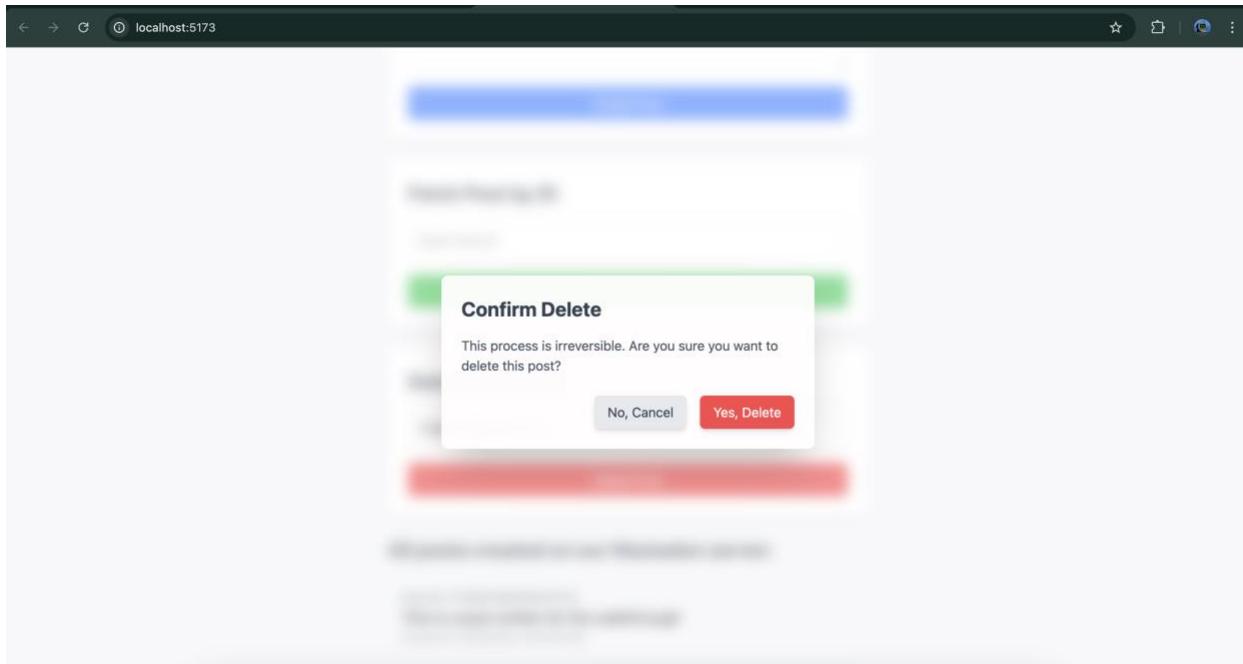
This function also takes the contents of the ID textbox as input. When the delete button is clicked, the handleDeleteClick() function is called which displays a modal modal prompting the user to confirm. When the user confirms the delete, the handleConfirmDelete() function is called which performs the delete operation. If the input is a valid ID of a post on the Mastodon server, this function assembles a DELETE request containing the Mastodon instance address, post ID to be deleted, authorization token, and content-type. This request deletes the post which the ID corresponds in our server and displays a success message on the UI. If the ID is invalid or doesn't exist, an failure message is displayed on the UI.

The walkthrough is as follows:

1. Provide an ID that you want to delete:

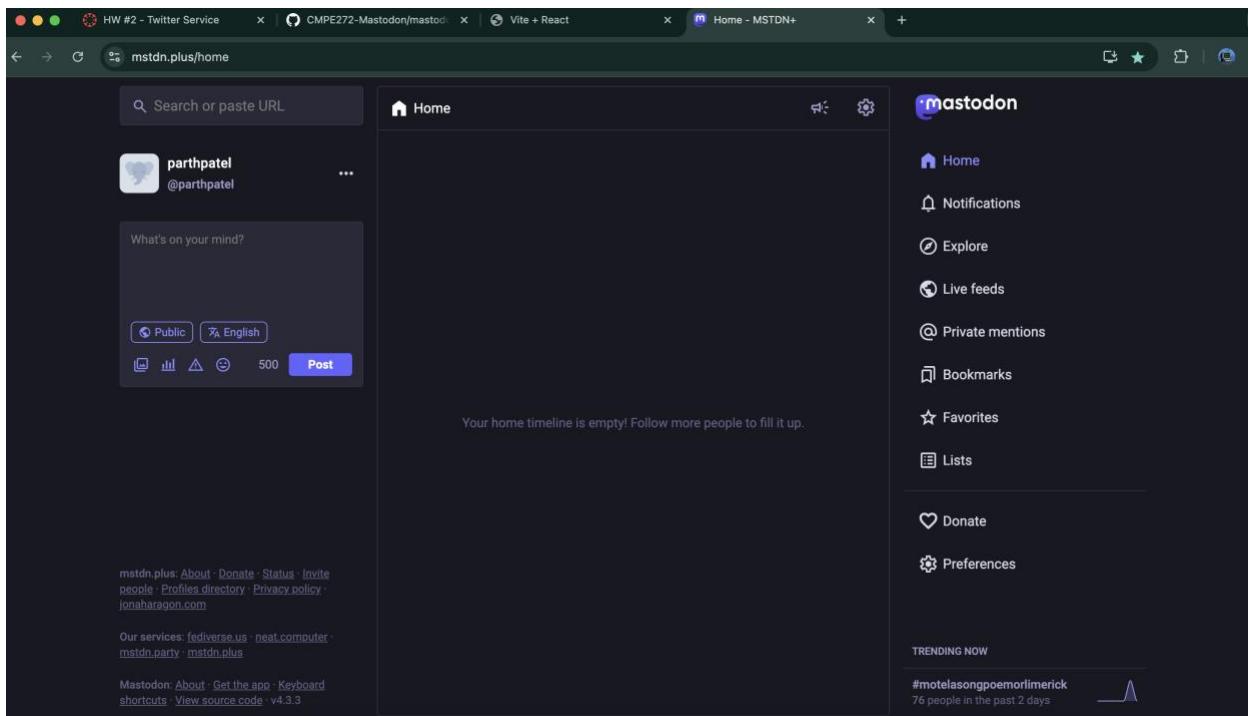


2. Click Yes, delete to confirm



A screenshot of a web application interface. At the top, there is a text input field with placeholder text "Write your post here..." and a blue "Create Post" button below it. Below this section is a white box labeled "Fetch Post by ID" containing an input field for "Enter Post ID" and a green "Fetch Post" button. Further down is another white box labeled "Delete Post by ID" containing an input field for "Enter Post ID" and a red "Delete Post" button. A green success message "Post deleted successfully! ID: 114062168305015113" is displayed below the delete button. At the bottom of the page, the text "All posts created on our Mastodon server:" is visible.

3. Confirm it has been removed on the server



Fetch All Posts

File: mastodon/src/App.tsx

Function: fetchAllPosts()

Implemented by: Parth Patel

Since we cannot share access to the Mastodon server directly, all the posts created are shown below. The function that performs this task is `fetchAllPosts()`.

Suppose 3 posts are created, this section would look like the following:

The screenshot shows a web application interface for interacting with a Mastodon server. At the top, there are two input fields: one for 'Enter Post ID' with a green 'Fetch Post' button, and another for 'Delete Post ID' with a red 'Delete Post' button. Below these are three sections listing posts:

- Post ID: 114065965645321044**
Real Post #3
Posted at: 2/25/2025, 10:35:08 AM
- Post ID: 114065965377780328**
Real Post #2
Posted at: 2/25/2025, 10:35:04 AM
- Post ID: 114065964886177063**
Real post #1
Posted at: 2/25/2025, 10:34:57 AM

The screenshot shows the mstdn.plus home page. On the left, there's a sidebar with a search bar and a profile section for 'parthpatel'. The main area displays a timeline of posts from the user:

- parthpatel** @parthpatel
Real Post #3 (posted 28s ago)
- parthpatel** @parthpatel
Real Post #2 (posted 32s ago)
- parthpatel** @parthpatel
Real post #1 (posted 39s ago)

Below the timeline, there are links for 'mstdn plus: About · Donate · Status · Invite people · Profiles directory · Privacy policy · Jonaharagon.com' and 'Our services: fediverse.us · neat_computer · masto.party · mstdn.plus'. On the right, a sidebar titled 'Mastodon' lists navigation options: Home, Notifications, Explore, Live feeds, Private mentions, Bookmarks, Favorites, Lists, Donate, and Preferences. It also shows a 'TRENDING NOW' section for '#tunetuesday'.