Concerns:

1. If I make a post, and come back the next day would I be able to delete it?

2. But how does it differentiate between users and recognize their posts?

   - When I was asked about how deletion works and differentiates between users, I initially said it should but wasn't entirely sure if it was implemented properly since I hadn't tested the system across multiple devices or browsers. I wanted to be careful not to claim something I hadn't verified.

Explanation:

- The logic was in fact functioning as intended, since the first two posts created during early testing couldn't be deleted from my current browser session, while newer posts showed the delete option. Therefore, confirming that the ownership logic was implemented correctly to begin with.

How it works:

- The project's foundational idea is to provide users with anonymity, while still giving ownership and control over what they've shared; and this was implemented by using a session-based system rather than user accounts.

- When a user visits the site, a unique token is assigned and stored in a cookie; and any post created is tagged in the database with that same token. That token persists across sessions and browser restarts. So if a user returns the next day on the same device and browser, the cookie is still present meaning the token still matches, and the delete option will appear for their posts. However, if the user switches devices, uses a different browser, clears their cookies, or opens the site in private/incognito mode, that token is lost and can not be deleted from that new session.

How it identifies browser session using a token stored in a cookie:

1) Open the site:

- GET / route calls **_get_or_set_delete_token()**
- If the browser doesn't already have a cookie named remains_token, the server generates one:

```
token = secrets.token_urlsafe(24)
```

- Then it sends it back as a cookie:

```
resp.set_cookie("remains_token", token, max_age=60*60*24*365,
samesite="Lax")
```

So from now on, that browser carries a "secret key" **(remains_token)**.

2) When post is created:

In **POST /api/posts**, it gets the token from the cookie and save it into the post document:

```
"delete_token": token
```

So every post "belongs" to the token that created it

3) When the wall loads, it decides if delete is allowed:

In **GET /api/posts**, the server compares:

- the post's stored **delete_token**
- the browser's current cookie **remains_token**

```
"can_delete": (p.get("delete_token") == token)
```

If they match → **can_delete = true** → JS shows the delete button

4) Server-side enforcement

In **DELETE /api/posts/<id>**, server checks that browser's stored token matches the token saved with the post and post is removed if both values align in the database:

```
posts_col.delete_one({"_id": oid, "delete_token": token})
```

So you can't delete other people's posts unless you literally have their cookie token.

What appeared to be a technical limitation and flaw, actually helped me validate the project's logic!