Homework 5

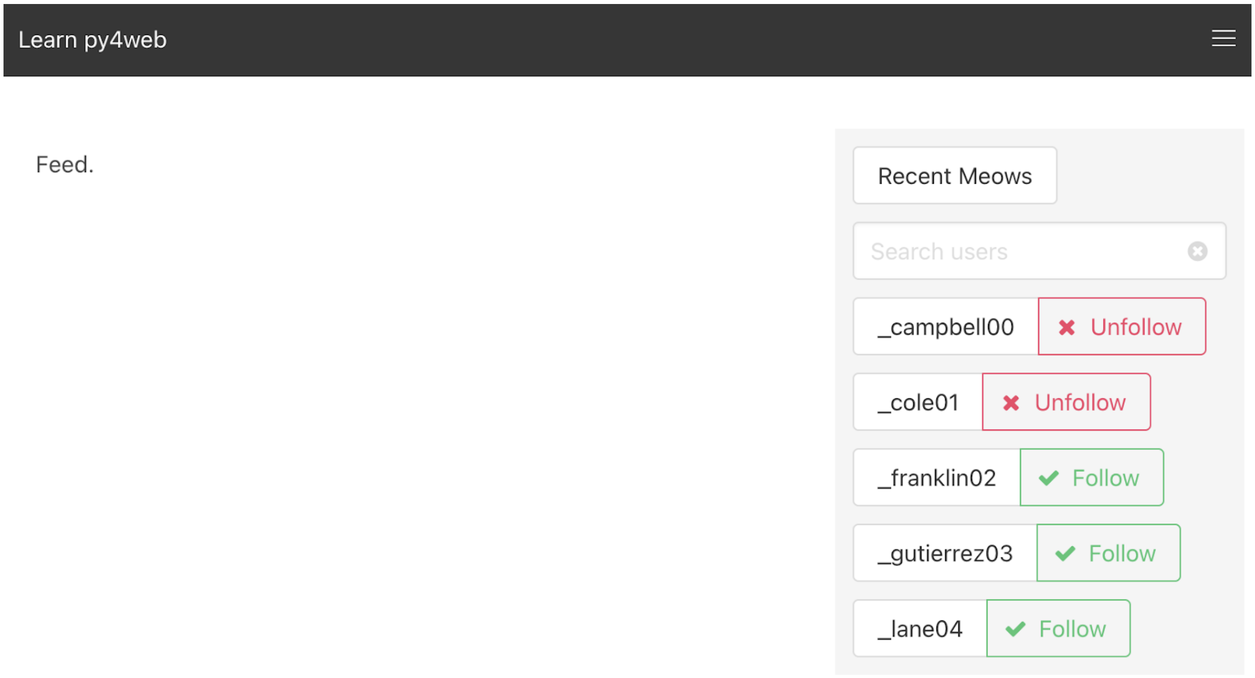
CSE 183, 2023

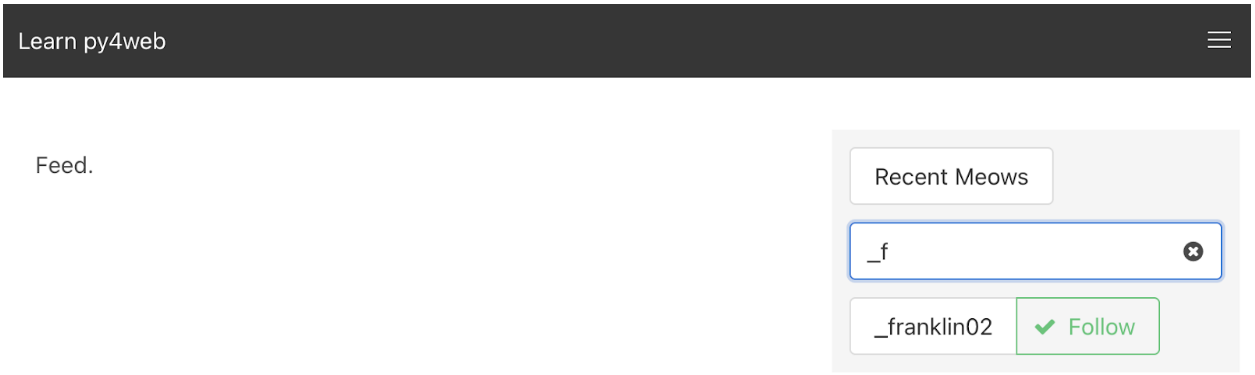
In this homework, you have to develop a Twitter-like site, where you can search users and follow/unfollow them, meaw (the equivalent of twit: cats eat birds), remeaw (the equivalent of retweet) and reply to a meaw.

We will guide you in the implementation step by step.

Step 1: User management

Before meows, we need users. We will build a site that enables us search users, and follow/unfollow them. The goal of step 1 is to build a site that looks like this:





You can search users, follow them, and unfollow them. Precisely:

- When you follow a user, the button changes to show “unfollow” (so you can unfollow them if you change your mind)
- When you unfollow a user, the button changes to show “follow”.
- When you search users, only users beginning with the given search string should appear listed.
- When you click on the x in the search users field, you should reload the users.

When reloading or loading the users:

- At most 20 users should be loaded.
- You should first load followed users; then if there is still space, non-followed users.

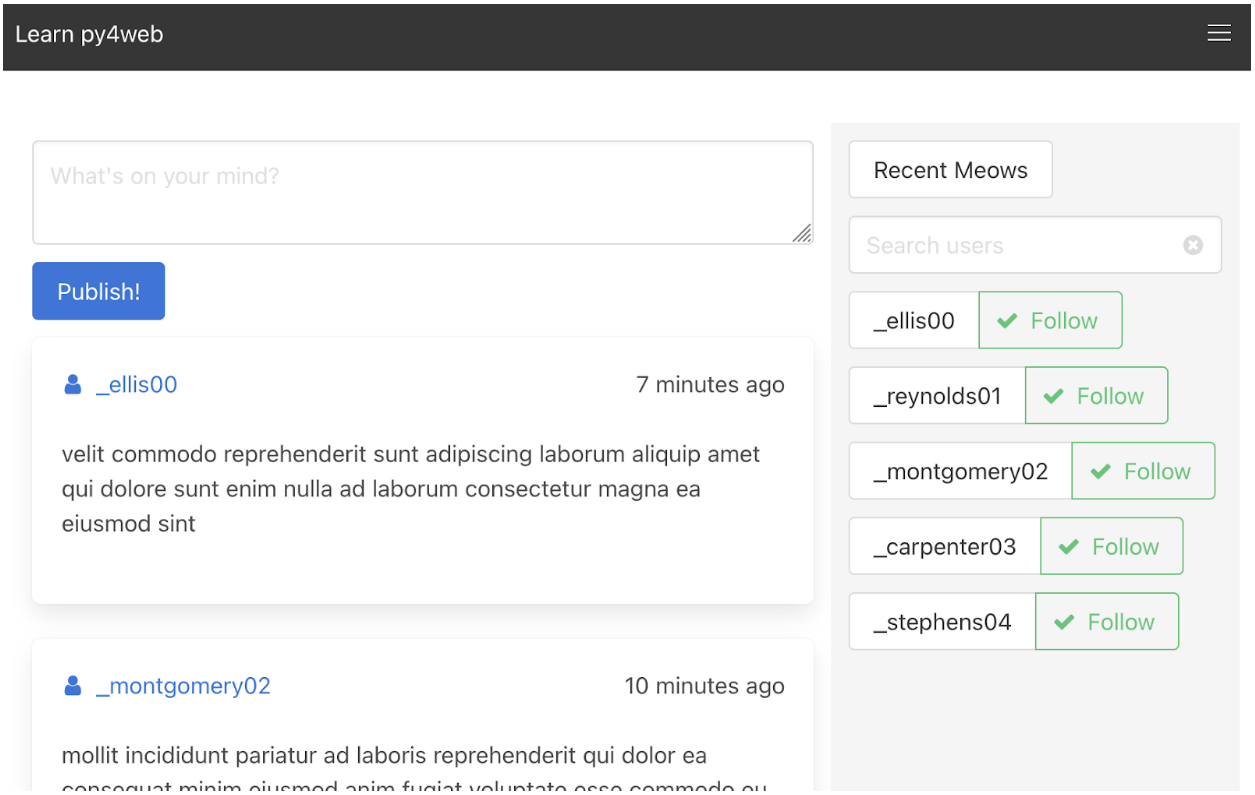
The “follow” is a many-to-many relation, and you should declare it in the database like this:

```
db.define_table('follow', ...)
```

You can use this application as the starting point of your work.

Step 2: Posting Meows (meowing?)

Your next step should be to allow the logged-in user to meow. Here is how the application should look like:



_ellis00 7 minutes ago

velit commodo reprehenderit sunt adipiscing laborum aliquip amet qui dolore sunt enim nulla ad laborum consectetur magna ea eiusmod sint

_montgomery02 10 minutes ago

mollit incididunt pariatur ad laboris reprehenderit qui dolor ea consequat minim eiusmod anim fugiat voluptate esse commodo eu

Below is some help for you.

Inserting test content

This code will insert both users and their meows in the database for testing. You can adapt the code to your needs.

```
def add_users_for_testing(num_users):
    # Test user names begin with "_".
    # Counts how many users we need to add.
    db(db.meow).delete()
    db(db.follow).delete()
    db(db.auth_user.username.startswith("_")).delete()
    num_test_users = db(db.auth_user.username.startswith("_")).count()
    num_new_users = num_users - num_test_users
    print("Adding", num_new_users, "users.")
    for k in range(num_test_users, num_users):
        first_name = random.choice(FIRST_NAMES)
        last_name = first_name = random.choice(LAST_NAMES)
        username = "_%s%.2i" % (first_name.lower(), k)
        user = dict(
            username=username,
            email=username + "@ucsc.edu",
            first_name=first_name,
            last_name=last_name,
            password=username,  # To facilitate testing.
        )
        auth.register(user, send=False)
        # Adds some content for each user.
        ts = datetime.datetime.utcnow()
        for n in range(3):
            ts -= datetime.timedelta(seconds=random.uniform(60,  1000))
            m = dict(
                author=username,
                timestamp = ts,
                content=" ".join(random.choices(list(IUP.keys()), k=20))
            )
            db.meow.insert(**m)
    db.commit()
```

The Feed

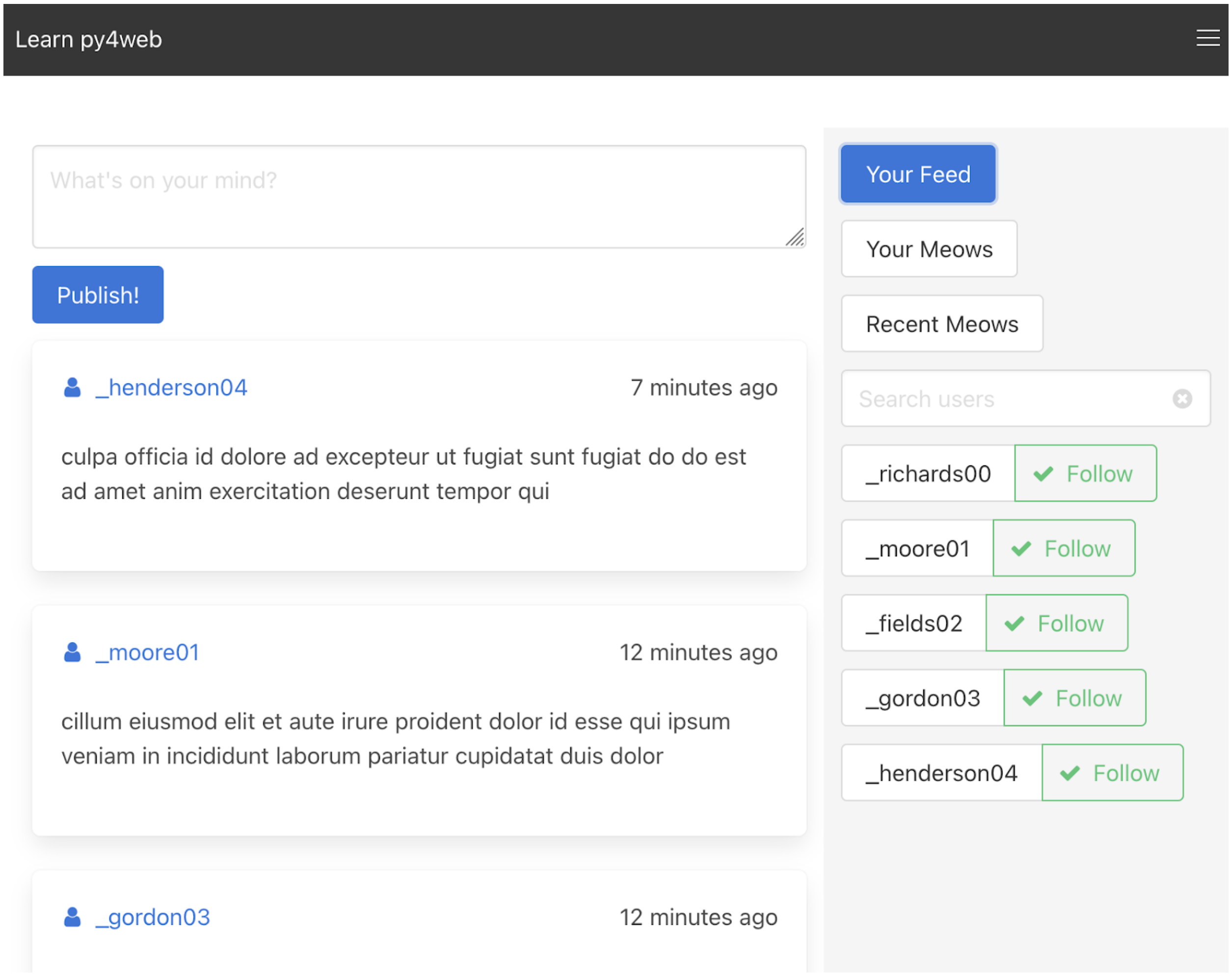
Below is HTML that creates the feed, without the Vue hooks you should insert.

```
<div class="field">
  <textarea class="textarea" placeholder="What's on your mind?" rows="2"></textarea>
</div>
<div class="field">
  <div class="control"><button class="button is-link">Publish!</button></div>
</div>

<div class="box">
  <nav class="level is-mobile">
    <div class="level-left">
      <a class="level-item" aria-label="reply">
        <span class="icon is-small">
          <i class="fa fa-user"></i>
        </span>
        <span class="ml-2">Grizabella</span>
      </a>
    </div>
    <div class="level-right">
      <span class="level-item">
        One minute ago
      </span>
    </div>
  </nav>
  <div class="content">
    I love tuna fish.
  </div>
  <nav class="level is-mobile">
    <div class="level-left">
      <a class="level-item" aria-label="reply">
        <span class="icon is-small">
          <i class="fa fa-reply" aria-hidden="true"></i>
        </span>
        <span class="ml-1">4</span>
      </a>
      <a class="level-item" aria-label="retweet">
        <span class="icon is-small">
          <i class="fa fa-retweet" aria-hidden="true"></i>
        </span>
      </a>
    </div>
  </nav>
</div>
```

3. Feed Selection

Your next step is to allow the user to select which feed to see. Here is how the application should look like:



_henderson04 7 minutes ago

culpa officia id dolore ad excepteur ut fugiat sunt fugiat do do est ad amet anim exercitation deserunt tempor qui

_moore01 12 minutes ago

cillum eiusmod elit et aute irure proident dolor id esse qui ipsum veniam in incididunt laborum pariatur cupidatat duis dolor

_gordon03 12 minutes ago

There are four kind of choices for the user:

- Your Feed: this displays the meows of the people followed by the user. If there are no such meows to display (e.g., if the person is not following anyone), this should display recent meows.
- Your Meows: this displays the meows of the user who is logged in.
- Recent Meows: this displays recent meows (by anyone).
- Clicking on a user on the right displays the meows by that user.

The active choice should be indicated in blue color: for example, above we are displaying Your Feed. You should display at most 20 meows for any choice. No dynamic loading required.

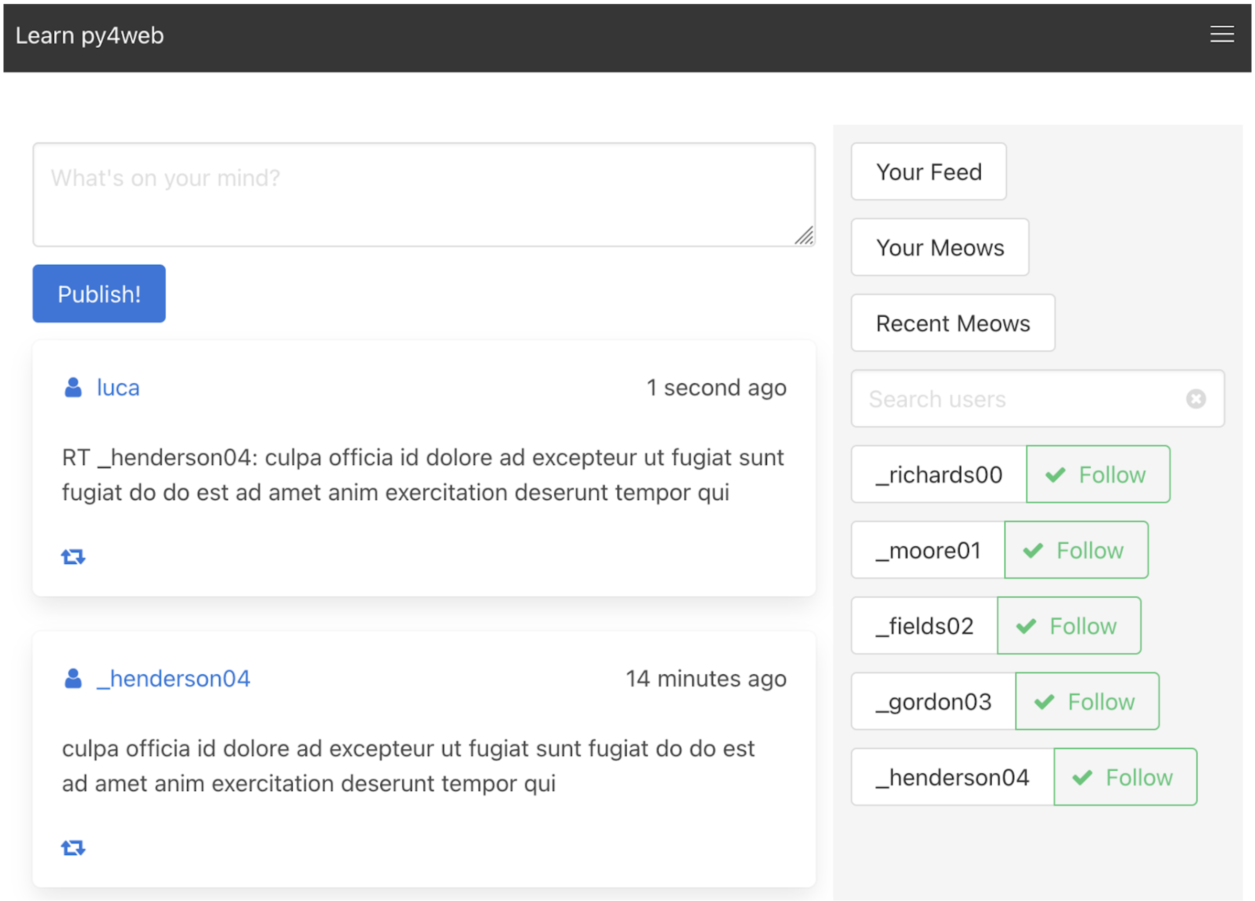
Note how time is displayed. This can be obtained via Python and via the Sugar Javascript library. In Python, be sure to return timestamps in isoformat, using `t.isoformat()` for a time t expressed in UTC time. In Javascript, you can do:

```
Sugar.Date(t + "Z").relative()
```

to format an isoformat string t as a relative time.

4. Retweets, pardon: Remeows

Next, you should enable the remeow feature. Below, user luca has just remeowed the meow by _henderson04.



luca 1 second ago

RT _henderson04: culpa officia id dolore ad excepteur ut fugiat sunt fugiat do do est ad amet anim exercitation deserunt tempor qui

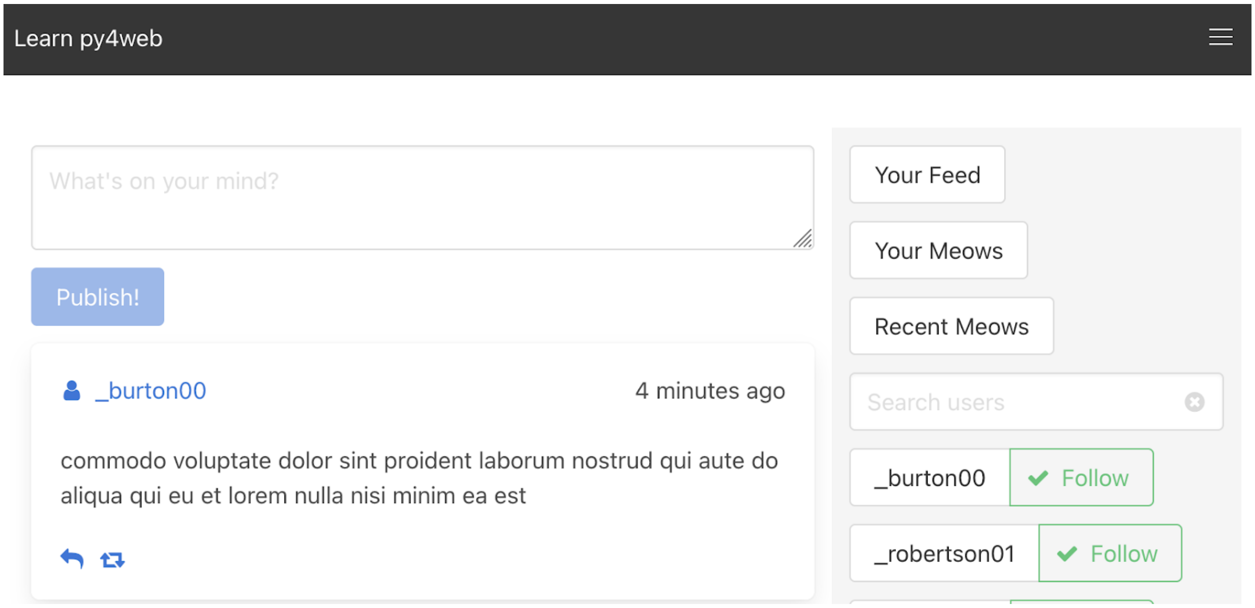
_henderson04 14 minutes ago

culpa officia id dolore ad excepteur ut fugiat sunt fugiat do do est ad amet anim exercitation deserunt tempor qui

Hint: try to reuse as much code as you can. You really do not need to add a lot of new code to get this to work.

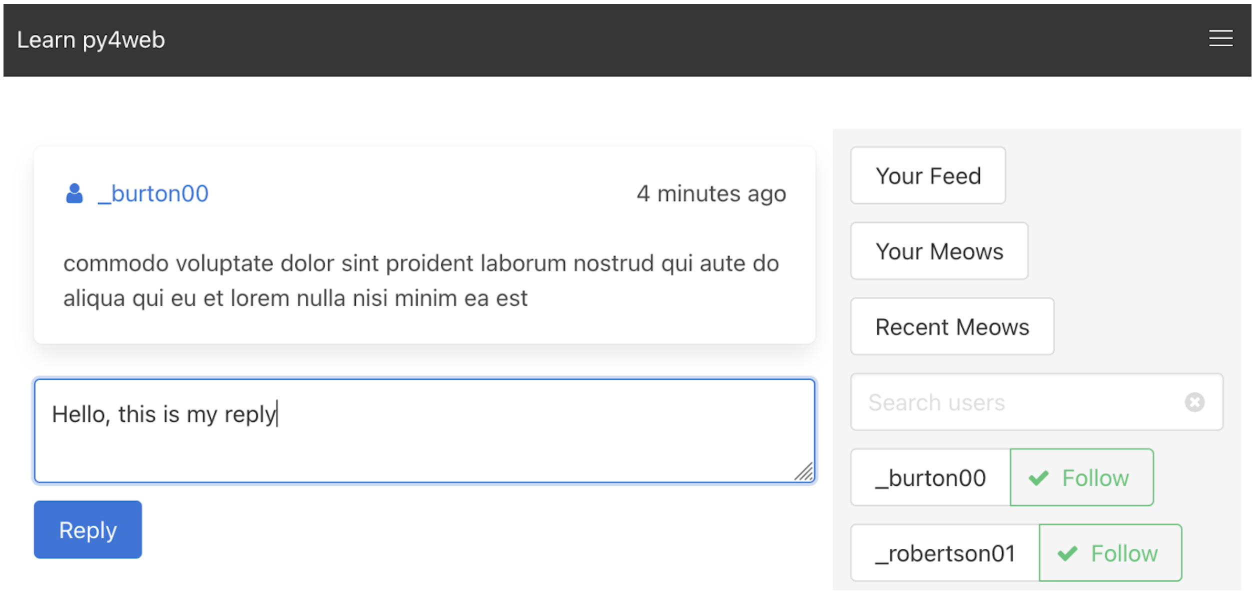
5. Replies

The last step consists in implementing replies. Suppose luca (the logged-in user) wants to reply to _burton00. Then luca should press on the reply button below:



commodo voluptate dolor sint proident laborum nostrud qui aute do aliqua qui eu et lorem nulla nisi minim ea est

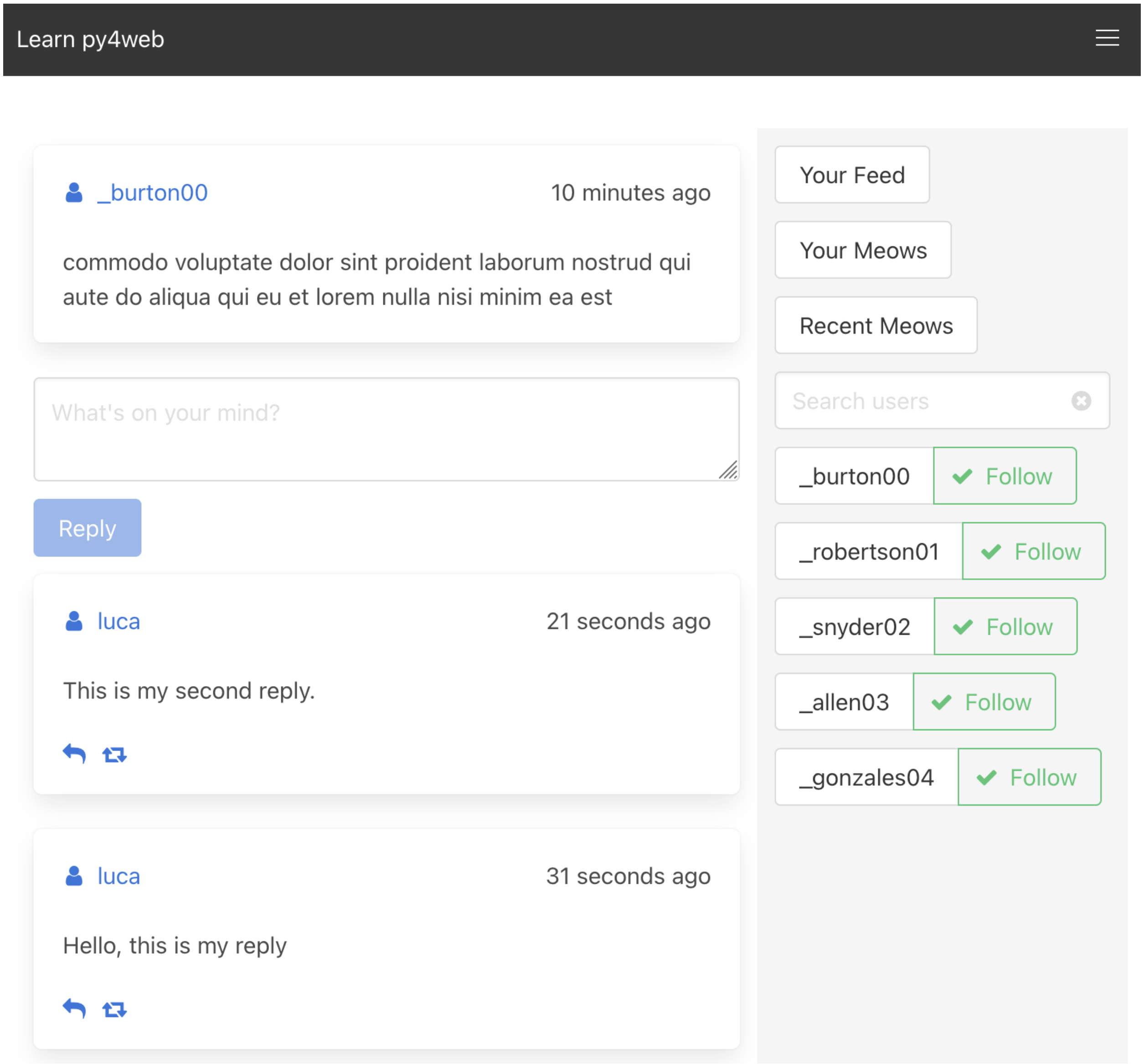
Without a screen reload (i.e., implementing this as a single-page app), this should show a feed like this:



commodo voluptate dolor sint proident laborum nostrud qui aute do aliqua qui eu et lorem nulla nisi minim ea est

Hello, this is my reply

As you can see, the meow to which one is replying is at the top. Next comes an input area for the reply. Below it go the replies; after luca types two replies, the reply view should look like this:



_burton00 10 minutes ago

commodo voluptate dolor sint proident laborum nostrud qui aute do aliqua qui eu et lorem nulla nisi minim ea est

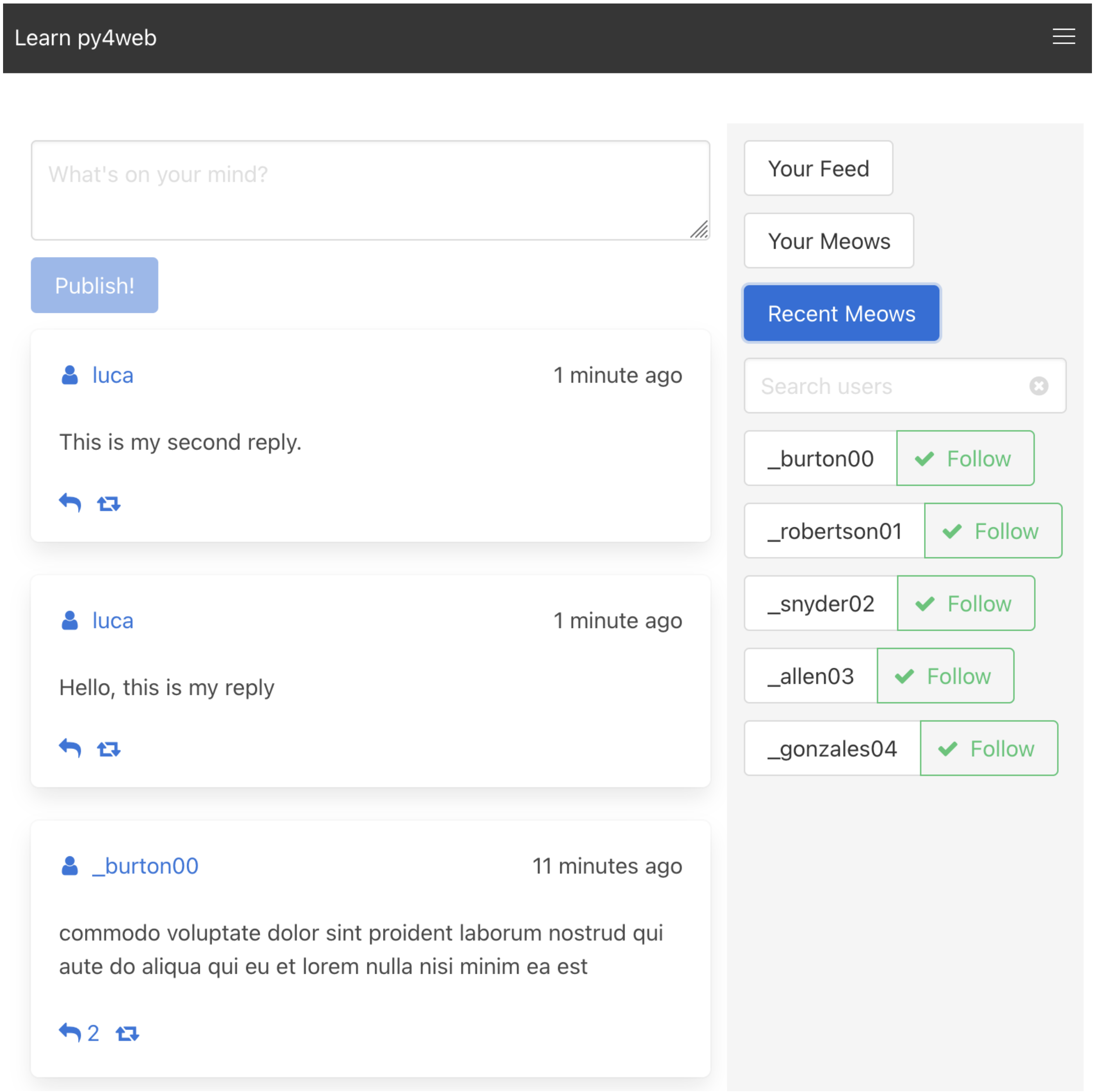
luca 21 seconds ago

This is my second reply.

luca 31 seconds ago

Hello, this is my reply

From here, clicking on any of the feed buttons leads back to the main feed view. For example, clicking on Recent Meows produces:



luca 1 minute ago

This is my second reply.

luca 1 minute ago

Hello, this is my reply

_burton00 11 minutes ago

commodo voluptate dolor sint proident laborum nostrud qui aute do aliqua qui eu et lorem nulla nisi minim ea est

2

Note how the meow by _burton00 now has 2 replies (the number 2 next to the reply icon).

Replies are displayed in the reply view, but are otherwise normal meows, which can be re-meowed, replied to in turn, etc., and appear in the feeds according to the normal rules.

That’s all!