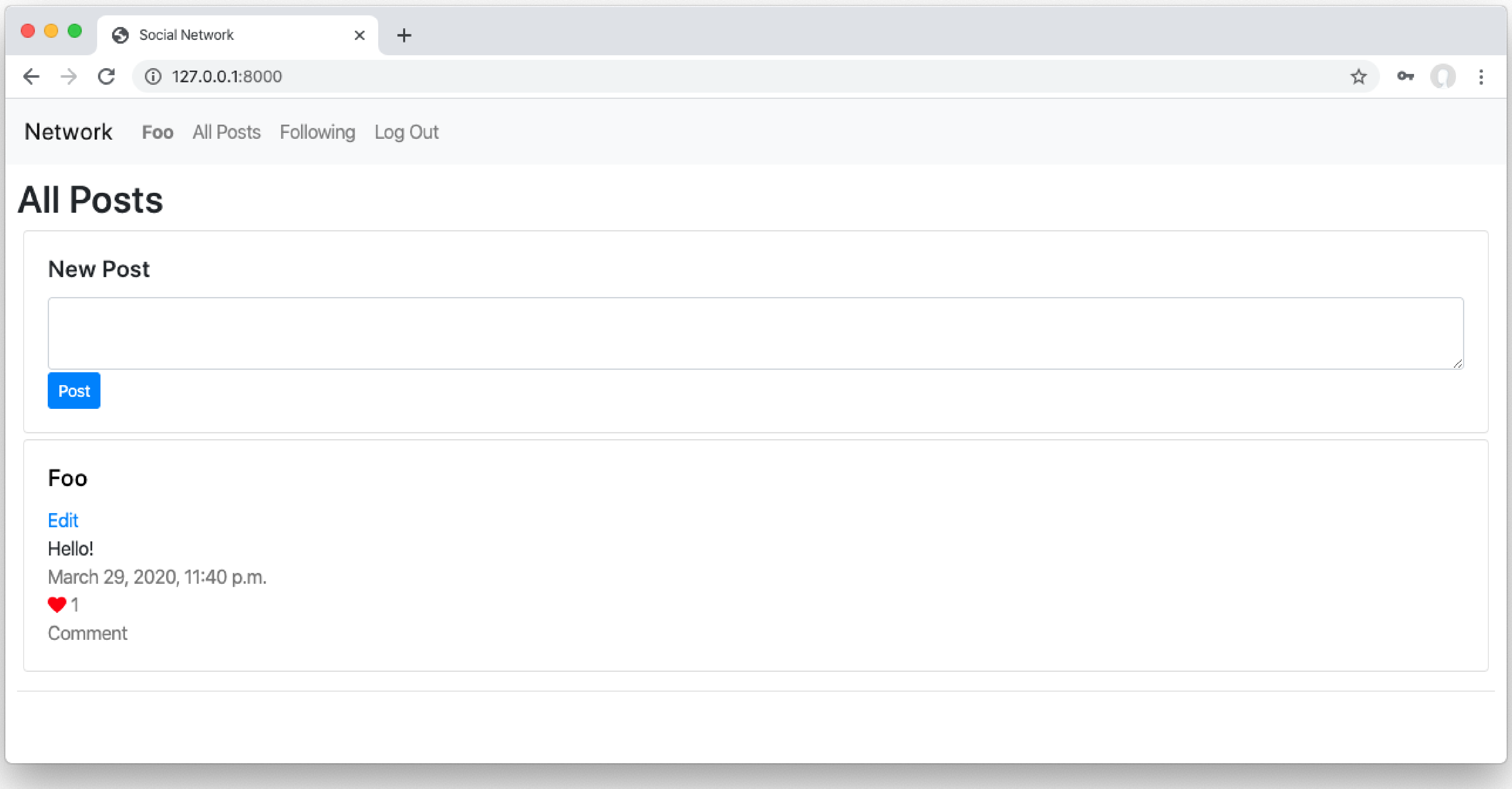
CSCI E-33a

Web Programming with Python and JavaScript

Harvard Extension School (https://www.extension.harvard.edu/)
Spring 2021

Network

Design a Twitter-like social network website for making posts and following users.



Getting Started

1. Download the distribution code from https://cdn.cs50.net/web/2020/spring/projects/4/network.zip (https://cdn.cs50.net/web/2020/spring/projects/4/network.zip) and unzip it.
2. In your terminal, `cd` into the `project4` directory.
3. Run `python manage.py makemigrations network` to make migrations for the `network` app.
4. Run `python manage.py migrate` to apply migrations to your database.

Understanding

In the distribution code is a Django project called `project4` that contains a single app called `network`, structured similarly to Project 2's `auctions` app.

First, open up `network/urls.py`, where the URL configuration for this app is defined. Notice that we've already written a few URLs for you, including a default index route, a `/login` route, a `/logout` route, and a `/register` route.

Take a look at `network/views.py` to see the views that are associated with each of these routes. The index view for now returns a mostly-empty `index.html` template. The `login_view` view renders a login form when a user tries to GET the page. When a user submits the form using the POST request method, the user is authenticated, logged in, and redirected to the index page. The `logout_view` view logs the user out and redirects them to the index page. Finally, the `register` route displays a registration form to the user, and creates a new user when the form is submitted. All of this is done for you in the distribution code, so you should be able to run the application now to create some users.

Run `python manage.py runserver` to start up the Django web server, and visit the website in your browser. Click "Register" and register for an account. You should see that you are now “Signed in as” your user account, and the links at the top of the page have changed. How did the HTML change? Take a look at `network/templates/network/layout.html` for the HTML layout of this application. Notice that several parts of the template are wrapped in a check for if `user.is_authenticated`, so that different content can be rendered depending on whether the user is signed in or not. You’re welcome to change this file if you’d like to add or modify anything in the layout!

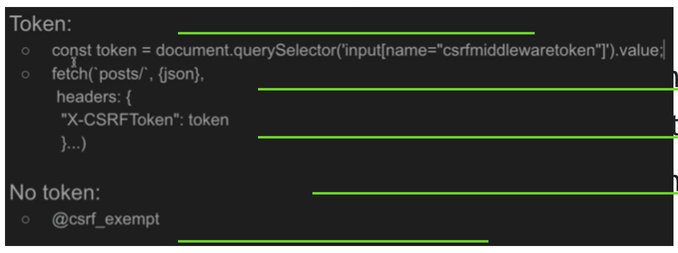
Finally, take a look at `network/models.py`. This is where you will define any models for your web application, where each model represents some type of data you want to store in your database. We’ve started you with a `User` model that represents each user of the application. Because it inherits from `AbstractUser`, it will already have fields for a username, email, password, etc., but you’re welcome to add new fields

to the `User` class if there is additional information about a user that you wish to represent. You will also need to add additional models to this file to represent details about posts, likes, and followers. Remember that each time you change anything in `network/models.py`, you’ll need to first run `python manage.py makemigrations` and then `python manage.py migrate` to migrate those changes to your database.

Specification

Using Python, JavaScript, HTML, and CSS, complete the implementation of a social network that allows users to make posts, follow other users, and “like” posts. You must fulfill the following requirements:

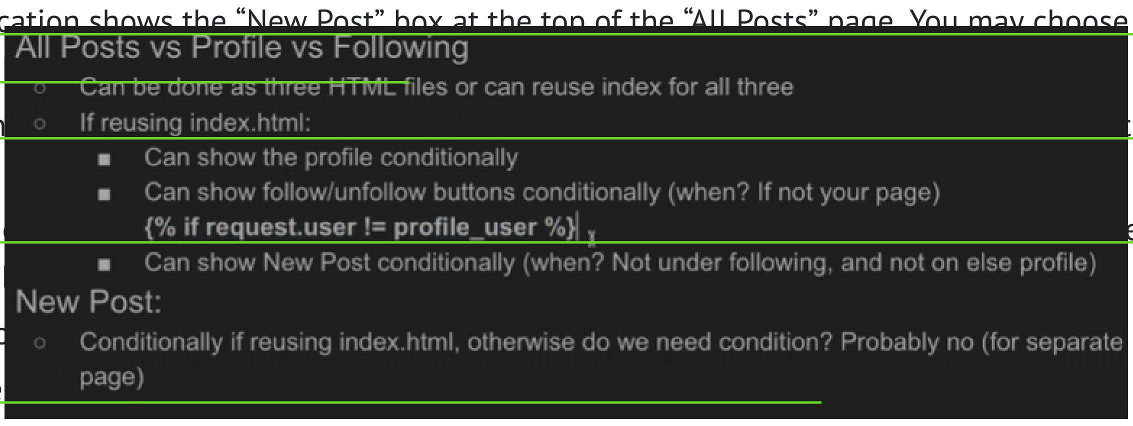
- **New Post**: Users who are signed in should be able to write a new text-based post by filling in text into a text area and then clicking a


Token:
```
○ const token = document.querySelector('input[name="csrfmiddlewaretoken"]').value;
○ fetch(`posts/`, {json},
   headers: {
    "X-CSRFToken": token
   }...)
```
No token:
```
○ @csrf_exempt
```

his specification shows the “New Post” box at the top of the “All Posts” page. You may choose to do

the “New

navigation th the


All Posts vs Profile vs Following
- Can be done as three HTML files or can reuse index for all three
- If reusing index.html:
  - Can show the profile conditionally
  - Can show follow/unfollow buttons conditionally (when? If not your page) **{% if request.user != profile_user %}**
  - Can show New Post conditionally (when? Not under following, and not on else profile)

New Post:
- Conditionally if reusing index.html, otherwise do we need condition? Probably no (for separate page)

  - Each post should include the username e, and the number of “likes” the post has (this will
- **Profile Page**: Clicking on a username should lo
  - Display the number of followers the use
  - Display all of the posts for that user, in reverse chronological order.
  - For any other user who is signed in, this page should also display a “Follow” or “Unfollow” button that will let the current user toggle whether or not they are following this user’s posts. Note that this only applies to any “other” user: a user should not be able to follow themselves.
- **Following**: The “Following” link in the navigation bar should take the user to a page where they see all posts made by users that the current user follows.
  - This page should behave just as the “All Posts” page does, just with a more limited set of posts.
  - This page should only be available to users who are signed in.
- **Pagination**: On any page that displays posts, posts should only be displayed 10 on a page. If there are more than ten posts, a “Next” button should appear to take the user to the next page of posts (which should be older than the current page of posts). If not on the first page, a “Previous” button should appear to take the user to the previous page of posts as well.
  - See the **Hints** section for some suggestions on how to implement this.
- **Edit Post**: Users should be able to click an “Edit” button or link on any of their own posts to edit that post.
  - When a user clicks “Edit” for one of their own posts, the content of their post should be replaced with a `textarea` where the user can edit the content of their post.
  - The user should then be able to “Save” the edited post. Using JavaScript, you should be able to achieve this without requiring a reload of the entire page.
  - For security, ensure that your application is designed such that it is not possible for a user, via any route, to edit another user’s posts.
- **“Like” and “Unlike”**: Users should be able to click a button or link on any post to toggle whether or not they “like” that post.
  - Using JavaScript, you should asynchronously let the server know to update the like count (as via a call to `fetch`) and then update the post’s like count displayed on the page, without requiring a reload of the entire page.

Hints

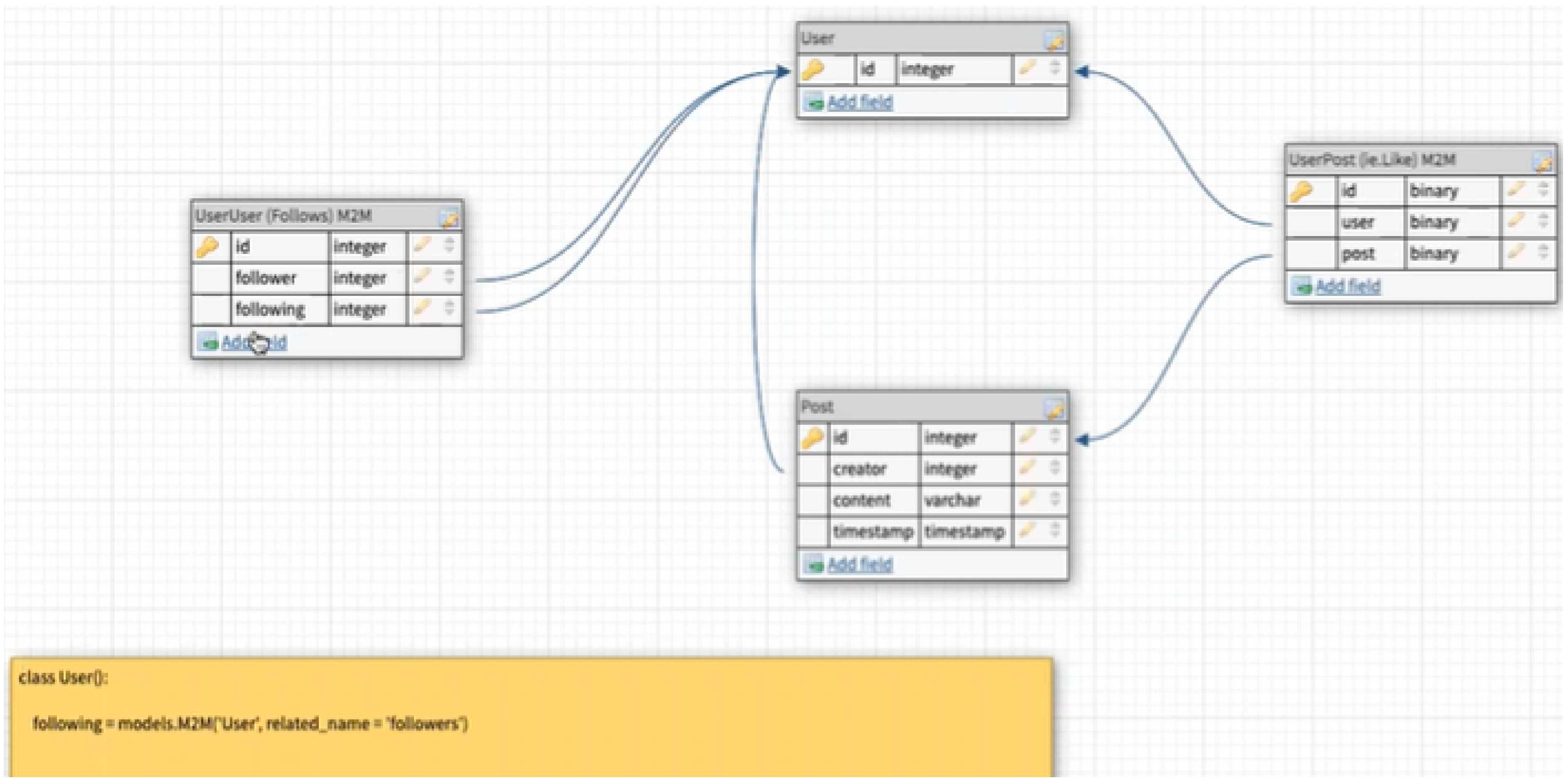
- For examples of JavaScript `fetch` calls, you may find some of the routes in Project 3 useful to reference.
- You’ll likely need to create one or more models in `network/models.py` and/or modify the existing `User` model to store the necessary data for your web application.
- Django’s Paginator (https://docs.djangoproject.com/en/3.0/topics/pagination/) class may be helpful for implementing pagination on the back-end (in your Python code).

- Bootstrap's Pagination (https://getbootstrap.com/docs/4.4/components/pagination/) features may be helpful for displaying pages on the front-end (in your HTML).
- You do not need to use React to complete this assignment but if you plan to use it on your final, this is a great opportunity to practice!

How to Submit

1. Download a ZIP file containing your implementation of this project.
2. Go to CSCI E-33a's Gradescope page (https://www.gradescope.com/courses/215635).
3. Click "Project 4: Network".
4. Drag and drop your ZIP file to the area that says "Drag & Drop".
5. Click **Upload**.

You should see a message that says "Project 4: Network submitted successfully!" Contact your teaching fellow if not!



```
class User():
  following = models.M2M('User', related_name = 'followers')
```