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 button to submit the post.
  + The screenshot at the top of this specification shows the “New Post” box at the top of the “All Posts” page. You may choose to do this as well, or you may make the “New Post” feature a separate page.
* **All Posts**: The “All Posts” link in the navigation bar should take the user to a page where they can see all posts from all users, with the most recent posts first.
  + Each post should include the username of the poster, the post content itself, the date and time at which the post was made, and the number of “likes” the post has (this will be 0 for all posts until you implement the ability to “like” a post later).
* **Profile Page**: Clicking on a username should load that user’s profile page. This page should:
  + Display the number of followers the user has, as well as the number of people that the user follows.
  + 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.