This list we’ve been working with currently has no provisions for assigning list items to specific users. Using the listapp obtained from the course resources in Reggienet and <https://docs.djangoproject.com/en/4.1/topics/auth/default/> as a reference implement the following user story:

As a list author I need to be able to associate list items with an account so that I can differentiate my list items from others’.

Step one: Verify needed Django apps are installed in your project settings:

<https://docs.djangoproject.com/en/4.1/topics/auth/>

<https://docs.djangoproject.com/en/4.1/topics/settings/>

Step 2: Edit your projects root url configuration to include django.contrib.auth.urls

                https://docs.djangoproject.com/en/4.0/topics/auth/default/#using-the-views

The source code for the auth app is available here: https://github.com/django/django/blob/main/django/contrib/auth

                Using this as a reference, what capabilities does this app provide for us?

 (hint: look at urls.py and views.py)

Step 3: Create templates.

                Create a folder for your templates (templates/registration) Looking at the source code foe the auth views (<https://github.com/django/django/blob/main/django/contrib/auth/views.py>) we can see that the login view, and others expects to find templates in a folder called ‘registration’ so it is important that we use this name.

Here is an example template for a login pager, courtesy of learndjango.com

|  |
| --- |
| <h2>Log In</h2>  <form method="post">    {% csrf\_token %}   {{ form.as\_p }}    <button type="submit">Log In</button>  </form> |

For your project to be able to find this template, you’ll need to add the path to your project settings. You’ll find a dictionary named TEMPLATES in your project settings. Add your templates to the dictionalry like this, again courtesy learndjango.com:

'DIRS': [BASE\_DIR / "templates"],

Now that the registration folder is in your templates dictionary, the auth app will be able to find its templates.

At this point you should have a working login page which you can view at /127.0.0.1:8000/accounts/login

You can provide custom templates for any of the other view in the same way. The auth app doesn’t include much in the way of templates, so you’ll need to write your own.

Step four: Associate your models with a user.

                For our items to belong to a user, we’ll need link them together. Since many items can belong to one user, we’ll use the many-to-one relationship. To do this you’ll need to import the user model from django contrib.auth and add it as a foreign key field:

<https://docs.djangoproject.com/en/4.1/topics/db/examples/many_to_one/>

Step five: add some users and items

                Since our app doesn’t yet have the capabilities to add users and items, we only have two other options for adding this data to our app, the python console, or the built in Django admin page. In this case, let’s use the admin app. First, you’ll need to create a super user. From the console, type “python manage.py createsuperuser” You’ll need to provide a username and password. Once you are done, open 127.0.0.1/admin and login with your credentials. Browse through the interface adding users and list items as you desire. The admin app is setup for use by default when you start a project, but if you want to learn more about it, read here:

https://docs.djangoproject.com/en/4.1/ref/contrib/admin/

Take a screenshot of your models showing some users and their list items in the admin app and submit for credit

On Wednesday we’ll work on updating the app to only show a logged in user’s items.