Lab 6 Part 1
Custom User Model

Django's built-in User model allows us to start working with users right away, as we just did with our Blog app in a previous lab. However, the official Django documentation highly recommends using a custom user model for new projects. The reason is that if you want to make any changes to the User model down the road—-for example adding an age field-—using a custom user model from the beginning makes this quite easy. But if you do not create a custom user model, updating the default User model in an existing Django project is very difficult.

Step 1 - Initial Set Up

1. Go to www.github.com and log in to your GitHub account.
2. Go to Moodle and click on the link for Lab 6 Upload Newspaper Application.
3. Once you have accepted the assignment you are asked to refresh the page, and you are then presented with a link to your repository for lab 6.
4. When you click on the repository link, you are taken to the repository where you see a readme file has already been added:
5. In Windows Command line make sure that you are in the **djangoprojects** folder and type the suitable command to activate the virtual environment:
6. Use the git clone command to clone the repo to your local computer:
7. Move into this **lab6-newspaper-application**.. folder using the **cd** command.
8. Create a new Django project called **newspaperproject** with the following command. Don't forget the period (fullstop) at the end:

```
django-admin startproject newspaperproject .
```

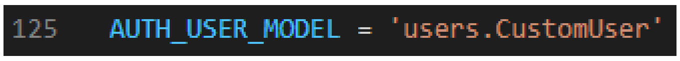
9. Create a new app called **users**
10. Open the project in VS Code and add this new app to the INSTALLED_APPS in **settings.py**

NOTE: DO NOT RUN THE MIGRATE COMMAND AT THIS STAGE

Step 2 - Custom User Model

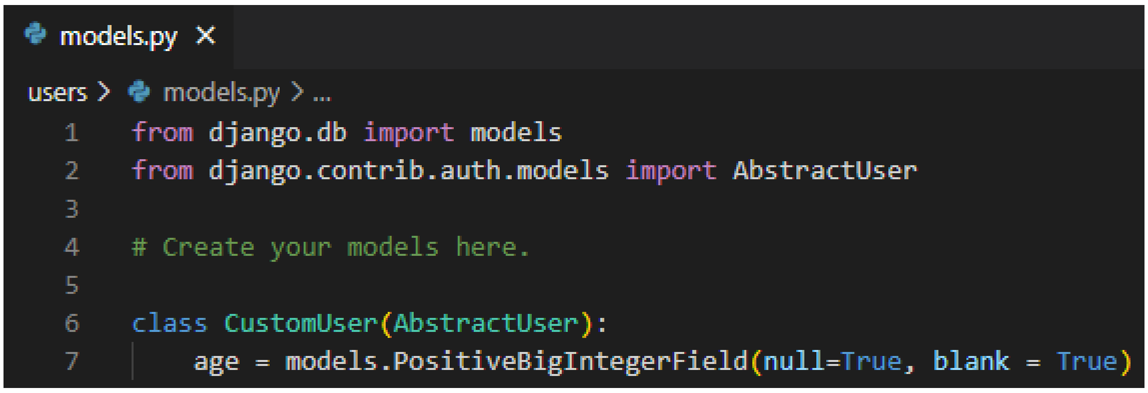
At the bottom of the **settings.py** file use the **AUTH_USER_MODEL** config to tell Django to use our new custom user model in place of the built-in User model. We will call our custom user model **CustomUser** so, since it exists within our users app we refer to it as **users.CustomUser**.

Add the following line of code at the end of the **settings.py** file:



```python
AUTH_USER_MODEL = 'users.CustomUser'
```

Update **users/models.py** with a new **User** model called **CustomUser** that extends the existing **AbstractUser**. We also include our first custom field, **age**, here.



models.py
users > models.py > ...

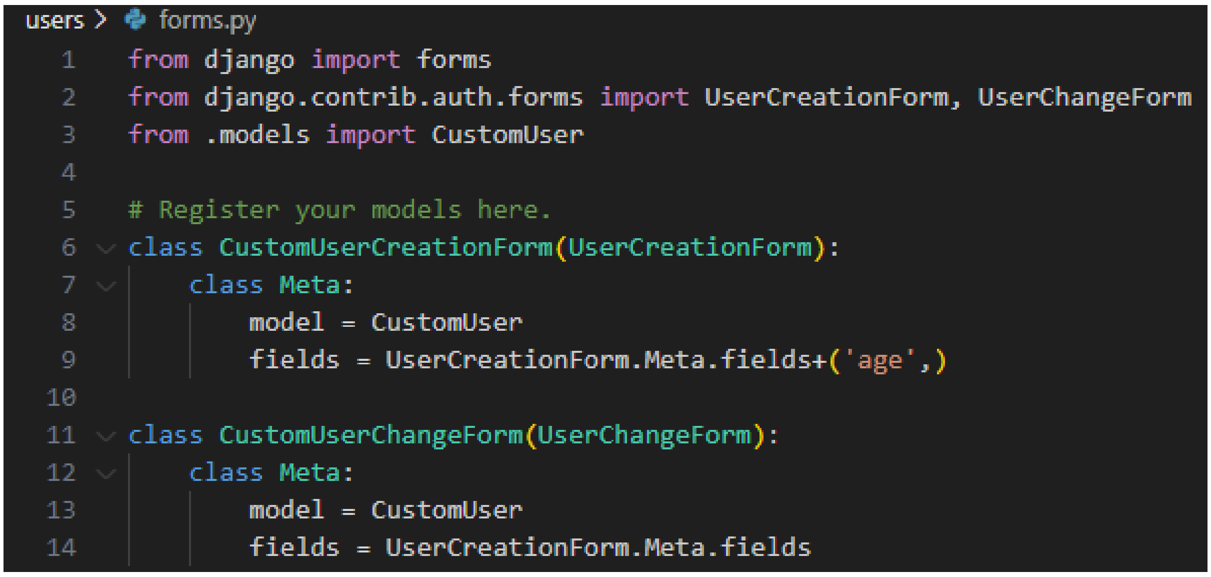
```python
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.

class CustomUser(AbstractUser):
    age = models.PositiveBigIntegerField(null=True, blank = True)
```

Step 3 - Forms

There are two ways in which we can interact with our new **CustomUser** model.

1. Within the admin app which allows us, as superusers, to modify existing users. We will need to update the two built-in forms for this functionality: **UserCreationForm** and **UserChangeForm**
2. When a user signs up for a new account on our website

Here we will focus on the **admin** app where we need to update the existing forms in the **admin** app to include the new field called **age**. Create a new file in the **users** app called **forms.py**. Update this file with the following code:


users > forms.py

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from .models import CustomUser

# Register your models here.
class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = CustomUser
        fields = UserCreationForm.Meta.fields+('age',)

class CustomUserChangeForm(UserChangeForm):
    class Meta:
        model = CustomUser
        fields = UserCreationForm.Meta.fields
```

For both new forms we are setting the model to our **CustomUser** and using the default fields via Meta.fields which includes all default fields. To add our custom age field, we simply tack it on at the end and it will display automatically on our future sign up page.

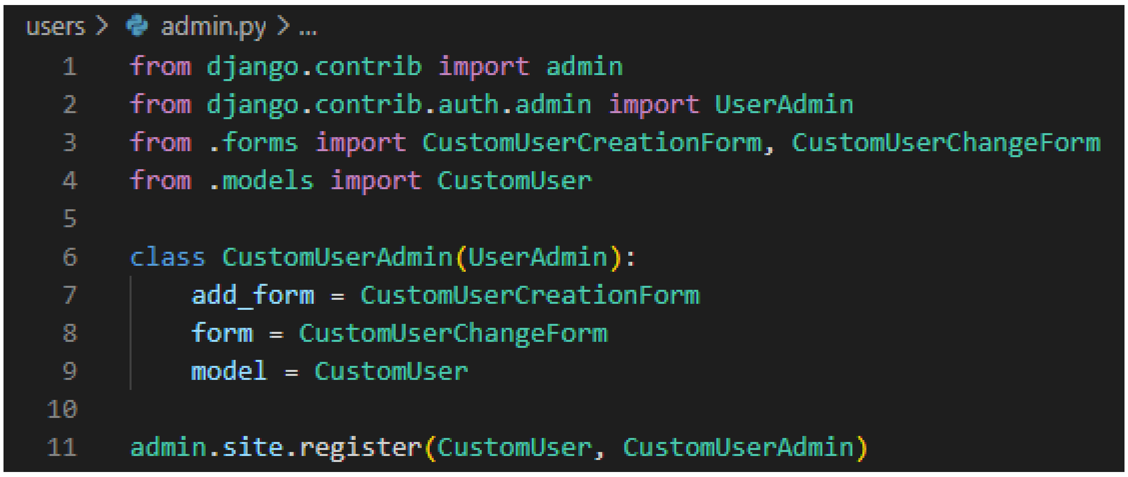
Our **CustomUser** model contains all the fields of the default User model and our additional age field which we set.

There are many default fields including username, first_name, last_name, email, password, groups, and more. Yet when a user signs up for a new account on Django the default form only asks for a username, email, and password. This tells us that the default setting for fields on **UserCreationForm** is just username, email, and password even though there are many more fields available.

Understanding forms and models properly takes some time. In the next exercise we will create our own sign up, log in, and log out pages which will tie together our **CustomUser** model and forms more clearly.

The only other step we need is to update our **admin.py** file since Admin is tightly coupled to the default **User** model. We will extend the existing **UserAdmin** class to use our new **CustomUser** model.

Open **admin.py** and enter the following code:



users > admin.py > ...

```python
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .forms import CustomUserCreationForm, CustomUserChangeForm
from .models import CustomUser

class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser

admin.site.register(CustomUser, CustomUserAdmin)
```

Run **makemigrations** and **migrate** for the first time to create a new database that uses the custom user model.

Command Line

---

python manage.py makemigrations users

---

python manage.py migrate

Superuser

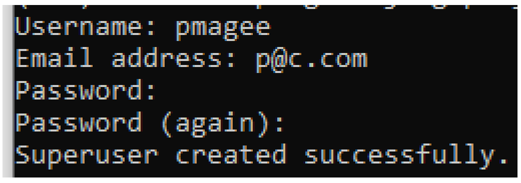
Create a superuser account to confirm that everything is working as expected. On the command line type the following command and go through the prompts.

Command Line

---

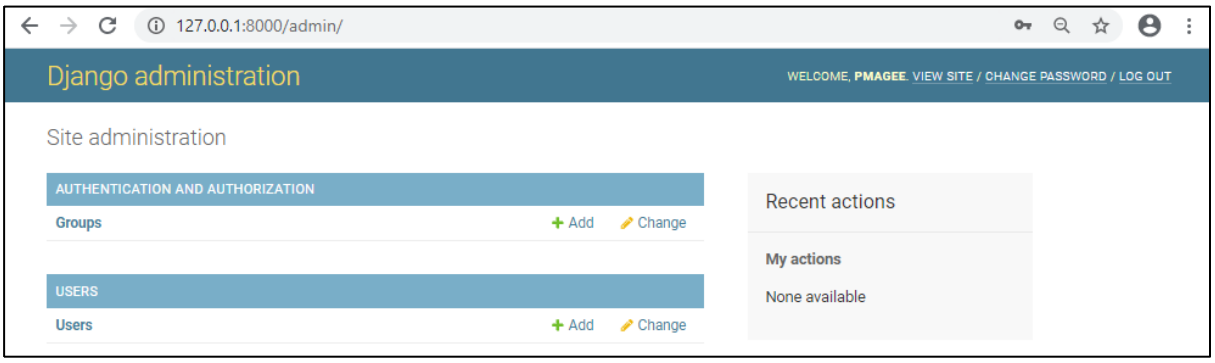
python manage.py createsuperuser

You should be able to create a new user which demonstrates that our custom user model works as expected.



```
Username: pmagee
Email address: p@c.com
Password:
Password (again):
Superuser created successfully.
```

Start up the web server and navigate to the admin at http://127.0.0.1:8080/admin and log in.



127.0.0.1:8000/admin/

Django administration

WELCOME, PMAGEE. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups + Add Change

USERS

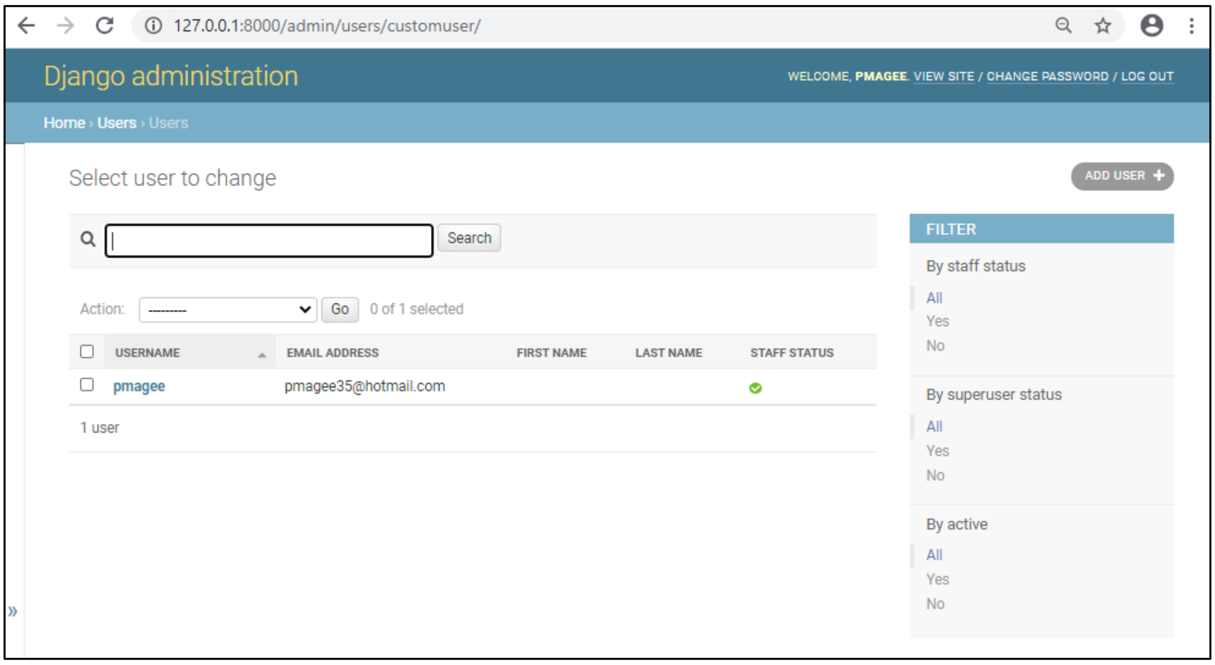
Users + Add Change

Recent actions

My actions

None available

If you click on the link for “**Users**” you should see your superuser account as well as the default fields of Username, Email Address, First Name, Last Name, and Staff Status but there is no age field here.



127.0.0.1:8000/admin/users/customuser/

Django administration

WELCOME, PMAGEE. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Users › Users

Select user to change

ADD USER +

Search

Action: --------- Go 0 of 1 selected

	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
☐	pmagee	pmagee35@hotmail.com			✓

1 user

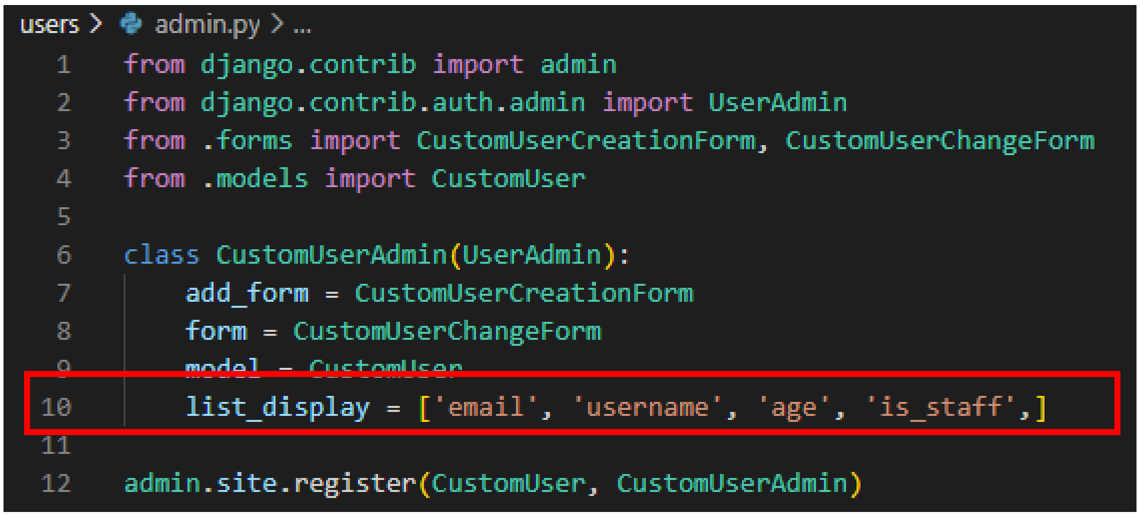
FILTER

By staff status
All
Yes
No

By superuser status
All
Yes
No

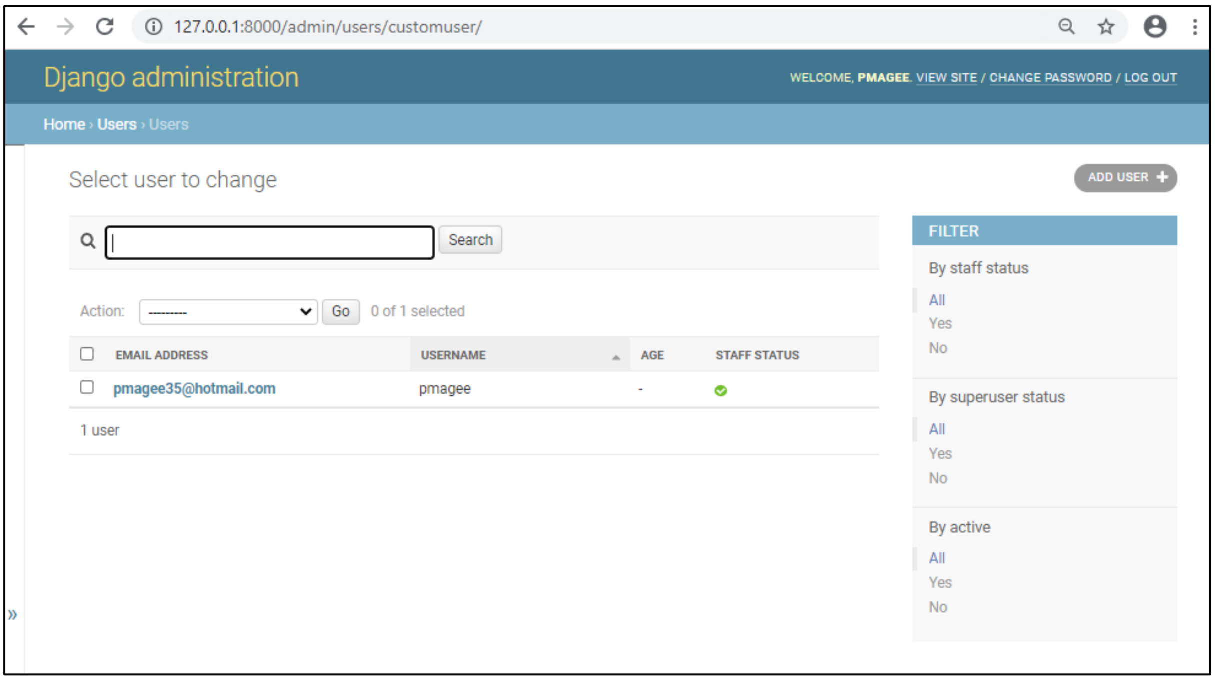
By active
All
Yes
No

We can control the fields listed here via the **list_display** setting for **CustomUserAdmin**. Add the following line of code to **admin.py**:


users > admin.py > ...

```python
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .forms import CustomUserCreationForm, CustomUserChangeForm
from .models import CustomUser

class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser
    list_display = ['email', 'username', 'age', 'is_staff',]

admin.site.register(CustomUser, CustomUserAdmin)
```

Refresh the page and you should see the update. Now we see the fields email, username, age and is_staff:


127.0.0.1:8000/admin/users/customuser/

Django administration

WELCOME, PMAGEE. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Users › Users

Select user to change

Action: --------- Go 0 of 1 selected

	EMAIL ADDRESS	USERNAME	AGE	STAFF STATUS
☐	pmagee35@hotmail.com	pmagee	-	✓

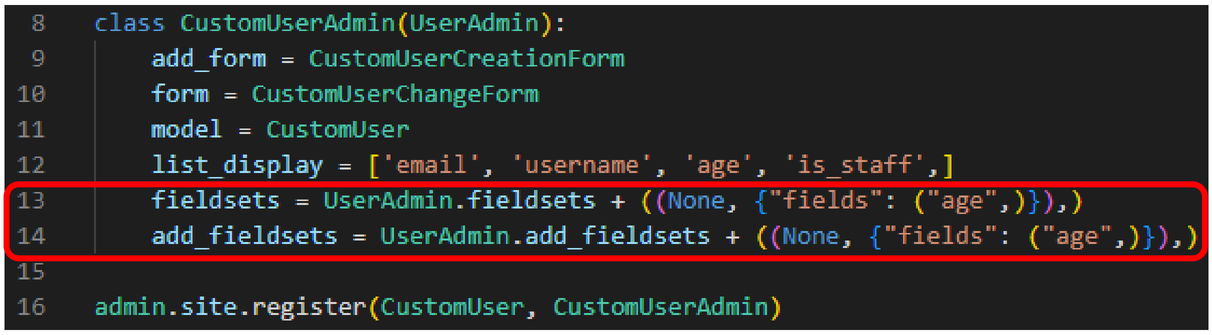
1 user

In Django Admin, click on the existing user and you are taken to an update page where you can update the user details but notice that there is no entry for our new **age** field that we added. Similarly, if we try to add a new user in Django Admin, the form we are presented with has form elements for the username and password but again none for the **age**.

To control the layout of these admin “add” and “change” pages, we use **fieldsets** and **addfieldsets**

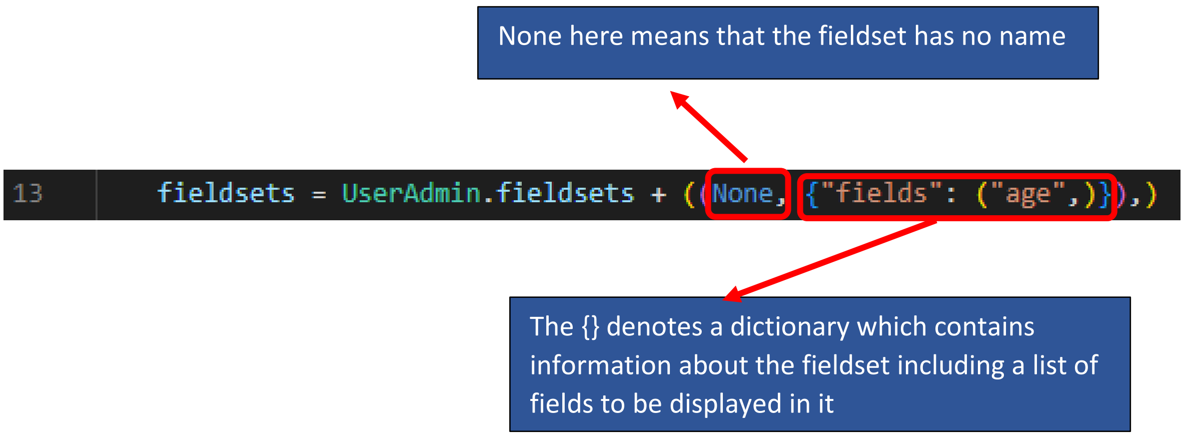
- **fieldsets** field is used to control the admin **change** user page
- **add_fieldsets** field allows us to control the layout of the admin **create** user page

Add the following two lines of code to **admin.py**:



```python
class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser
    list_display = ['email', 'username', 'age', 'is_staff',]
    fieldsets = UserAdmin.fieldsets + ((None, {"fields": ("age",)}),)
    add_fieldsets = UserAdmin.add_fieldsets + ((None, {"fields": ("age",)}),)

admin.site.register(CustomUser, CustomUserAdmin)
```

The **fieldsets** consist of a **list** of two-tuples, in which each two-tuple represents a <fieldset> on the admin form page. (A <fieldset> is a “section” of the form.)



None here means that the fieldset has no name

```python
    fieldsets = UserAdmin.fieldsets + ((None, {"fields": ("age",)}),)
```

The {} denotes a dictionary which contains information about the fieldset including a list of fields to be displayed in it

The **addfieldsets** composition is the same as above except that it is used to control the layout of the admin create user object page.

Save your project and go back into Django Admin and do the following:

- Add a new user with the age field now included (also include the email address)
- Edit an existing user with the age field included

Upload

1. Open the **README.md** file in Vs Code and replace the contents with your name and id number.
2. Save a snapshot of the current project state with the following command:
   git add -A
3. Commit the changes along with a suitable message:
   git commit -m “lab 6 part 1 commit”
4. Update the remote repository with the local commits:
   git push -u origin main
5. Go to your GitHub page and refresh the page to see your local code now hosted online.
6. To deactivate the virtual environment type, deactivate as shown below:
   deactivate
7. To exit out of Windows Command Line type exit:
   exit