Create a Registration functionality for the Author/User.

```
import re
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.models import User
from .models import Author


def User_register(request):
   if request.method == 'POST':
      # 1) grab & trim inputs
      username       = request.POST.get('username', '').strip()
      email          = request.POST.get('email', '').strip()
      password       = request.POST.get('password', '')
      confirm_password= request.POST.get('confirm_password', '')

      errors = {}

      # 2) basic non-empty checks
      if not username:
         errors['username'] = "Username can't be empty."
      if not email:
         errors['email'] = "Email can't be empty."
      if not password:
         errors['password'] = "Password can't be empty."
      if not confirm_password:
         errors['confirm_password'] = "Please confirm your password."

      # 3) email format
```

```
email_re = r'^[\w\.-]+@[\w\.-]+\.\w+$'
if email and not re.match(email_re, email):
   errors['email'] = "Enter a valid email address."

# 4) password match
if password and confirm_password and password != confirm_password:
   errors['confirm_password'] = "Passwords do not match."

# 5) uniqueness checks
if username and User.objects.filter(username=username).exists():
   errors['username'] = "That username is already taken."
if email and User.objects.filter(email=email).exists():
   errors['email'] = "That email is already registered."

# 6) if any errors, re-render form with messages
if errors:
   return render(request, 'registration/register.html', {
      'errors': errors,
      'form_values': {
         'username': username,
         'email': email,
      }
   })

# 7) all good ⇒ create User and Author
user = User.objects.create_user(username=username,
                        email=email,
                        password=password)
```

```
        # assume Author has a OneToOneField to User
        Author.objects.create(user=user)

        messages.success(request, "Registration successful! Please log in.")
        return redirect('login')   # or wherever your login URL is

    # GET → just show empty form
    return render(request, 'registration/register.html')
```

What you need in addition:

URL conf (urls.py):

```
from django.urls import path
from . import views

urlpatterns = [
    path('register/', views.User_register, name='register'),
    # ... other URLs ...
]
```

Template (templates/registration/register.html):

```
<h1>Sign Up</h1>
<form method="post">
 {% csrf_token %}
  <div>
   <label>Username:</label>
   <input type="text" name="username" value="{{ form_values.username|default:'' }}">
   <span style="color:red">{{ errors.username }}</span>
  </div>
  <div>
```

```
  <label>Email:</label>
  <input type="email" name="email" value="{{ form_values.email|default:'' }}">
  <span style="color:red">{{ errors.email }}</span>
 </div>
 <div>
  <label>Password:</label>
  <input type="password" name="password">
  <span style="color:red">{{ errors.password }}</span>
 </div>
 <div>
  <label>Confirm Password:</label>
  <input type="password" name="confirm_password">
  <span style="color:red">{{ errors.confirm_password }}</span>
 </div>
 <button type="submit">Register</button>
</form>
```

With this in place, your User_register() view will:

Verify none of the fields are blank

Check email against a simple regex

Ensure passwords match

Prevent duplicate usernames/emails

Create both the built-in User and your custom Author in one request