**📘 Django Custom Authentication System (FBV)**

We are building a **Custom Authentication System** in Django (using Function Based Views).  
This system covers **registration, login, logout, password reset, and password change.**

**🔹 1. Custom User Model Setup**

**account/models.py**

from django.db import models

from django.contrib.auth.models import BaseUserManager, AbstractBaseUser

class UserManger(BaseUserManager):

    def create\_user(self, email, password=None):

        # Creates and saves a User with the given email and password.

        if not email:

            raise ValueError("User must have a valid email address")

        user = self.model(email=self.normalize\_email(email))

        user.set\_password(password)

        user.save(using=self.\_db)

        return user

    def create\_superuser(self, email, password=None, \*\*extra\_fields):

        extra\_fields.setdefault('is\_staff', True)

        extra\_fields.setdefault('is\_superuser', True)

        if extra\_fields.get('is\_staff') is not True:

            raise ValueError('SuperUser must have is\_staff=True.')

        if extra\_fields.get('is\_superuser') is not True:

            raise ValueError('SuperUser must have is\_superuser=True.')

        user = self.create\_user(email, password)

        user.is\_staff = True

        user.is\_superuser = True

        user.is\_customer = True

        user.is\_seller = True

        user.save(using=self.\_db)

        return user

class User(AbstractBaseUser):

    email = models.EmailField(max\_length=255, unique=True)

    name = models.CharField(max\_length=255, blank=True)

    city = models.CharField(max\_length=255, blank=True)

    is\_active = models.BooleanField(default=False)

    is\_staff = models.BooleanField(default=False)

    is\_superuser = models.BooleanField(default=False)

    is\_customer = models.BooleanField(default=True)

    is\_seller = models.BooleanField(default=False)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    updated\_at = models.DateTimeField(auto\_now=True)

    USERNAME\_FIELD = 'email'  # login with email instead of username

    objects = UserManger()

    def \_\_str\_\_(self):

        return self.email

    def has\_perm(self, perm, obj=None):

        return self.is\_superuser

    def has\_module\_perms(self, app\_label):

        return self.is\_superuser

**project/settings.py**

# Custom User Model

AUTH\_USER\_MODEL = 'account.User'

**Steps**

1. Run migrations:
2. python manage.py makemigrations
3. python manage.py migrate
4. Database now has account\_user (not auth\_user).
5. Create superuser:
6. python manage.py createsuperuser

→ Superuser login with **email** instead of username.

1. Manually activate user (in DB is\_active=1) for testing.

**🔹 2. Registration System**

**account/utils.py**

from django.core.mail import EmailMultiAlternatives

from django.template.loader import render\_to\_string

from django.utils.html import strip\_tags

from django.conf import settings

import threading

# Threading for faster email sending

class SendEmailThread(threading.Thread):

    def \_\_init\_\_(self, email):

        self.email = email

        threading.Thread.\_\_init\_\_(self)

    def run(self):

        self.email.send()

def send\_activation\_email(recipient\_email, activation\_url):

    subject = "Activate your account on " + settings.SITE\_NAME

    from\_email = 'no\_reply@demomailtrap.com'

    to\_email = [recipient\_email]

    html\_content = render\_to\_string(

        'account/activation\_email.html', {'activation\_url': activation\_url})

    text\_content = strip\_tags(html\_content)

    email = EmailMultiAlternatives(subject, text\_content, from\_email, to\_email)

    email.attach\_alternative(html\_content, 'text/html')

    SendEmailThread(email).start()

def send\_reset\_password\_email(recipient\_email, reset\_url):

    subject = 'Reset Your Password on ' + settings.SITE\_NAME

    from\_email = 'no\_reply@demomailtrap.com'

    to\_email = [recipient\_email]

    html\_content = render\_to\_string(

        'account/reset\_password\_email.html', {'reset\_url': reset\_url})

    text\_content = strip\_tags(html\_content)

    email = EmailMultiAlternatives(subject, text\_content, from\_email, to\_email)

    email.attach\_alternative(html\_content, "text/html")

    SendEmailThread(email).start()

**settings.py (Email Config)**

SITE\_DOMAIN = 'http://127.0.0.1:8000/'

SITE\_NAME = 'Auth System'

EMAIL\_BACKEND = "django.core.mail.backends.smtp.EmailBackend"

EMAIL\_HOST = 'live.smtp.mailtrap.io'

EMAIL\_HOST\_USER = 'api'

EMAIL\_HOST\_PASSWORD = '21b5b11c16d665d4847fe6cb1141c06d'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

**🔹 3. Login System**

**account/views.py**

def login\_view(request):

    if request.user.is\_authenticated:

        if request.user.is\_seller:

            return redirect('seller\_dashboard')

        elif request.user.is\_customer:

            return redirect('customer\_dashboard')

        return redirect('home')

    if request.method == "POST":

        email = request.POST.get("email")

        password = request.POST.get("password")

        if not email or not password:

            messages.error(request, "Both fields are required.")

            return redirect('login')

        try:

            user = User.objects.get(email=email)

        except User.DoesNotExist:

            messages.error(request, "Invalid email or password.")

            return redirect('login')

        if not user.is\_active:

            messages.error(request, "Your account is inactive. Please activate your account.")

            return redirect('login')

        user = authenticate(request, email=email, password=password)

        if user is not None:

            login(request, user)

            if user.is\_seller:

                return redirect('seller\_dashboard')

            elif user.is\_customer:

                return redirect('customer\_dashboard')

            else:

                messages.error(request, "You do not have permission to access this area.")

                return redirect('home')

        else:

            messages.error(request, "Invalid email or password.")

            return redirect('login')

    return render(request, 'account/login.html')

**Logout**

# urls.py

from django.contrib.auth.views import LogoutView

urlpatterns = [

    path('logout/', LogoutView.as\_view(), name='logout'),

]

# settings.py

LOGOUT\_REDIRECT\_URL = 'login'

LOGIN\_URL = 'login'

**🔹 4. Password Change (Customer App)**

**customer/views.py**

from django.contrib.auth.forms import PasswordChangeForm

from django.contrib.auth.decorators import login\_required

@login\_required

def customer\_dashboard\_view(request):

    return render(request, 'customer/dashboard.html')

@login\_required

def password\_change\_view(request):

    if request.method == "POST":

        form = PasswordChangeForm(user=request.user, data=request.POST)

        if form.is\_valid():

            form.save()

            logout(request)

            messages.success(

                request, "Password changed successfully. Please log in with your new password."

            )

            return redirect('login')

        else:

            for field, errors in form.errors.items():

                for error in errors:

                    messages.error(request, error)

    else:

        form = PasswordChangeForm(user=request.user)

    return render(request, 'customer/password\_change.html', {'form': form})

**customer/password\_change.html**

<form method="POST">

    {% csrf\_token %}

    <input type="password" name="old\_password" placeholder="Old Password" required>

    <input type="password" name="new\_password1" placeholder="New Password" required>

    <input type="password" name="new\_password2" placeholder="Confirm Password" required>

    <button type="submit">Change Password</button>

</form>

**🔹 5. Password Reset**

**forms.py**

class PasswordResetForm(forms.Form):

    email = forms.EmailField(

        max\_length=255,

        required=True,

        widget=forms.EmailInput(attrs={'placeholder': 'you@example.com'})

    )

    def clean\_email(self):

        email = self.cleaned\_data.get('email')

        if not User.objects.filter(email=email).exists():

            raise forms.ValidationError(

                'No account is associated with this email address.'

            )

        return email

**settings.py**

# Password Reset Link Expiry

PASSWORD\_RESET\_TIMEOUT = 3600  # 1 hour

**password\_reset\_confirm.html**

<form method="POST">

    {% csrf\_token %}

    <input type="password" name="new\_password1" placeholder="New Password" required>

    <input type="password" name="new\_password2" placeholder="Confirm Password" required>

    <button type="submit">Reset Password</button>

</form>

**✅ Key Points**

* **Custom User Model** → Email as username.
* **Registration** → User gets activation email.
* **Login** → Redirects based on role (customer, seller).
* **Logout** → Uses Django LogoutView.
* **Password Change** → Uses PasswordChangeForm.
* **Password Reset** → Email link + expiry time.
* **Templates** must use correct input name attributes:
  + password\_reset\_confirm.html → new\_password1, new\_password2
  + customer/password\_change.html → old\_password, new\_password1, new\_password2

Complete Codes

# project/settings.py

"""

Django settings for ch42 project.

Generated by 'django-admin startproject' using Django 5.2.5.

Docs:

- Deployment checklist: https://docs.djangoproject.com/en/5.2/howto/deployment/

- Settings ref: https://docs.djangoproject.com/en/5.2/ref/settings/

"""

from pathlib import Path

# --- Paths -------------------------------------------------------------------

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# --- Security (dev defaults; change in production) ---------------------------

# ⚠️ Do NOT commit real secrets to Git. Use environment variables in prod.

SECRET\_KEY = 'django-insecure-c@8r8nmfokf#q\_bnpcvi7)d#qr4-ol989@#cprsaej=#my^t@s'

DEBUG = True

ALLOWED\_HOSTS = []  # e.g. ['yourdomain.com'] in production

# --- Installed Apps ----------------------------------------------------------

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',        # Auth framework (we override the user model)

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    # Local apps

    'customer',

    'seller',

    'account',

]

# --- Middleware --------------------------------------------------------------

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',              # CSRF protection

    'django.contrib.auth.middleware.AuthenticationMiddleware', # attaches request.user

    'django.contrib.messages.middleware.MessageMiddleware',    # messages framework

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'ch42.urls'

# --- Templates ---------------------------------------------------------------

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        # If you add a global "templates" folder at project root, put it here.

        'DIRS': [],

        'APP\_DIRS': True,  # auto-discover templates inside each app

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'ch42.wsgi.application'

# --- Database ----------------------------------------------------------------

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',      # Simple file DB for dev

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# --- Password validation (Django’s built-ins) --------------------------------

AUTH\_PASSWORD\_VALIDATORS = [

    {'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',},

    {'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',},

    {'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',},

    {'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',},

]

# --- I18N / TZ ---------------------------------------------------------------

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# --- Static files ------------------------------------------------------------

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

# --- Custom User Model -------------------------------------------------------

# Tell Django to use our email-first user model

AUTH\_USER\_MODEL = 'account.User'

# --- Site meta used in emails & links ----------------------------------------

SITE\_DOMAIN = 'http://127.0.0.1:8000/'  # local dev domain

SITE\_NAME = 'Auth System'

# --- Email (Mailtrap SMTP) ---------------------------------------------------

# ⚠️ In production, use env vars for secrets.

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'live.smtp.mailtrap.io'

EMAIL\_HOST\_USER = 'api'

EMAIL\_HOST\_PASSWORD = '21b5b11c16d665d4847fe6cb1141c06d'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

# --- Auth redirects -----------------------------------------------------------

LOGOUT\_REDIRECT\_URL = 'login'

LOGIN\_URL = 'login'

# --- Password reset token expiry (seconds) -----------------------------------

PASSWORD\_RESET\_TIMEOUT = 3600  # 1 hour

# project/urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    # Auth app (home/login/register/password reset)

    path('', include('account.urls')),

    # Role-based dashboards

    path('customer/', include('customer.urls')),

    path('seller/', include('seller.urls')),

]

# account/models.py

from django.db import models

from django.contrib.auth.models import BaseUserManager, AbstractBaseUser

class UserManger(BaseUserManager):

    """

    Custom manager to create users with email as the unique identifier.

    """

    def create\_user(self, email, password=None):

        """

        Creates and saves a User with the given email and password.

        """

        if not email:

            raise ValueError("User must have a valid email address")

        # normalize\_email lowercases the domain part

        user = self.model(email=self.normalize\_email(email))

        user.set\_password(password)  # hashes the raw password

        user.save(using=self.\_db)

        return user

    def create\_superuser(self, email, password=None, \*\*extra\_fields):

        """

        Creates and saves a Superuser with the given email and password.

        We force is\_staff and is\_superuser True for admin access.

        """

        extra\_fields.setdefault('is\_staff', True)

        extra\_fields.setdefault('is\_superuser', True)

        if extra\_fields.get('is\_staff') is not True:

            raise ValueError('SuperUser must have is\_staff=True.')

        if extra\_fields.get('is\_superuser') is not True:

            raise ValueError('SuperUser must have is\_superuser=True.')

        user = self.create\_user(email, password)

        # Optional project-specific roles

        user.is\_staff = True

        user.is\_superuser = True

        user.is\_customer = True

        user.is\_seller = True

        user.save(using=self.\_db)

        return user

class User(AbstractBaseUser):

    """

    Minimal custom user model using email instead of username.

    AbstractBaseUser provides: password, last\_login, is\_active (we override), etc.

    """

    email = models.EmailField(max\_length=255, unique=True)

    name = models.CharField(max\_length=255)              # make blank=True if optional

    city = models.CharField(max\_length=255)              # make blank=True if optional

    # Activation/roles

    is\_active = models.BooleanField(default=False)       # must manually activate or via link

    is\_staff = models.BooleanField(default=False)        # admin site access

    is\_superuser = models.BooleanField(default=False)    # full permissions

    is\_customer = models.BooleanField(default=True)      # role flag

    is\_seller = models.BooleanField(default=False)       # role flag

    # Timestamps

    created\_at = models.DateTimeField(auto\_now\_add=True) # when created

    updated\_at = models.DateTimeField(auto\_now=True)     # last updated

    # Use email for authentication instead of username

    USERNAME\_FIELD = 'email'

    # Attach custom manager

    objects = UserManger()

    def \_\_str\_\_(self):

        return self.email

    # Permissions: keep it simple for this project (superuser = all perms)

    def has\_perm(self, perm, obj=None):

        return self.is\_superuser

    def has\_module\_perms(self, app\_label):

        return self.is\_superuser

# account/forms.py

from django import forms

from account.models import User

class RegistrationForm(forms.ModelForm):

    """

    Simple registration form:

    - email, name from the model

    - password + confirm\_password are plain fields here; we hash password in the view

    """

    password = forms.CharField(widget=forms.PasswordInput)

    confirm\_password = forms.CharField(widget=forms.PasswordInput)

    class Meta:

        model = User

        fields = ["email", "name", "password", "confirm\_password"]

    # validate that password and confirm\_password match

    def clean(self):

        cleaned\_data = super().clean()

        password = cleaned\_data.get('password')

        confirm = cleaned\_data.get('confirm\_password')

        if password and confirm and password != confirm:

            self.add\_error('confirm\_password', 'Password and Confirm Password do not match.')

        return cleaned\_data

    # ensure email uniqueness at the form level

    def clean\_email(self):

        email = self.cleaned\_data.get('email')

        if email and User.objects.filter(email=email).exists():

            raise forms.ValidationError("A user with this email already exists.")

        return email

class PasswordResetForm(forms.Form):

    """

    Minimal 'forgot password' form that only requires an email.

    """

    email = forms.EmailField(

        max\_length=255,

        required=True,

        widget=forms.EmailInput(attrs={'placeholder': 'you@example.com'})

    )

    def clean\_email(self):

        """

        Ensure we only proceed if a user with this email actually exists.

        """

        email = self.cleaned\_data.get('email')

        if not User.objects.filter(email=email).exists():

            raise forms.ValidationError('No account is associated with this email address.')

        return email

# account/views.py

from django.shortcuts import render, redirect

from django.contrib import messages

from django.conf import settings

from django.utils.http import urlsafe\_base64\_encode, urlsafe\_base64\_decode

from django.utils.encoding import force\_bytes, force\_str

from django.contrib.auth.tokens import default\_token\_generator

from django.urls import reverse

from django.contrib.auth import authenticate, login

from django.contrib.auth.forms import SetPasswordForm

from account.forms import RegistrationForm, PasswordResetForm

from account.utils import send\_activation\_email, send\_reset\_password\_email

from account.models import User

def home(request):

    """Simple landing page."""

    return render(request, 'account/home.html')

def login\_view(request):

    """

    Email+password login.

    - If already authenticated, redirect based on role.

    - If POST: validate credentials and login.

    - Only active users can log in.

    """

    if request.user.is\_authenticated:

        if request.user.is\_seller:

            return redirect('seller\_dashboard')

        elif request.user.is\_customer:

            return redirect('customer\_dashboard')

        return redirect('home')

    if request.method == "POST":

        email = request.POST.get("email")

        password = request.POST.get("password")

        # simple required check

        if not email or not password:

            messages.error(request, "Both fields are required.")

            return redirect('login')

        # if email not found, avoid leaking which field is wrong

        try:

            user = User.objects.get(email=email)

        except User.DoesNotExist:

            messages.error(request, "Invalid email or password.")

            return redirect('login')

        # require activation first

        if not user.is\_active:

            messages.error(request, "Your account is inactive. Please activate your account.")

            return redirect('login')

        # authenticate using the email backend (we passed email to authenticate)

        user = authenticate(request, email=email, password=password)

        if user is not None:

            login(request, user)

            # role-based redirect

            if user.is\_seller:

                return redirect('seller\_dashboard')

            elif user.is\_customer:

                return redirect('customer\_dashboard')

            # fallback

            messages.error(request, "You do not have permission to access this area.")

            return redirect('home')

        # wrong credentials

        messages.error(request, "Invalid email or password.")

        return redirect('login')

    # GET -> show login form

    return render(request, 'account/login.html')

def register\_view(request):

    """

    Creates an inactive user; sends activation link via email.

    """

    if request.method == 'POST':

        form = RegistrationForm(request.POST)

        if form.is\_valid():

            # Build user but do not save password as plain text

            user = form.save(commit=False)

            user.set\_password(form.cleaned\_data["password"])

            user.is\_active = False

            user.save()

            # Build activation link (uid + token)

            uidb64 = urlsafe\_base64\_encode(force\_bytes(user.pk))

            token = default\_token\_generator.make\_token(user)

            activation\_link = reverse('activate', kwargs={'uidb64': uidb64, 'token': token})

            activation\_url = f'{settings.SITE\_DOMAIN}{activation\_link}'

            # Send activation email

            send\_activation\_email(user.email, activation\_url)

            messages.success(request, 'Registration successful! Please check your email to activate your account.')

            return redirect('login')

    else:

        form = RegistrationForm()

    return render(request, 'account/register.html', {'form': form})

def activate\_account(request, uidb64, token):

    """

    Validates activation token and activates the user.

    """

    try:

        uid = force\_str(urlsafe\_base64\_decode(uidb64))

        user = User.objects.get(pk=uid)

        # Already active?

        if user.is\_active:

            messages.warning(request, "This account has already been activated.")

            return redirect('login')

        # Validate the token and activate

        if default\_token\_generator.check\_token(user, token):

            user.is\_active = True

            user.save()

            messages.success(request, "Your account has been activated successfully.")

            return redirect('login')

        # If token invalid

        messages.error(request, "The activation link is invalid or has expired.")

        return redirect('login')

    except (TypeError, ValueError, OverflowError, User.DoesNotExist):

        messages.error(request, "Invalid activation link.")

        return redirect('login')

def password\_reset\_view(request):

    """

    Step 1 of 'Forgot Password':

    - Accept email, generate reset token, and send reset link.

    """

    if request.method == "POST":

        form = PasswordResetForm(request.POST)

        if form.is\_valid():

            email = form.cleaned\_data.get('email')

            user = User.objects.filter(email=email).first()

            if user:

                uidb64 = urlsafe\_base64\_encode(force\_bytes(user.pk))

                token = default\_token\_generator.make\_token(user)

                reset\_url = reverse('password\_reset\_confirm', kwargs={'uidb64': uidb64, 'token': token})

                absolute\_reset\_url = f"{request.build\_absolute\_uri(reset\_url)}"

                send\_reset\_password\_email(user.email, absolute\_reset\_url)

            # Always show success to avoid leaking user existence

            messages.success(request, "We have sent you a password reset link. Please check your email.")

            return redirect('login')

    else:

        form = PasswordResetForm()

    return render(request, 'account/password\_reset.html', {'form': form})

def password\_reset\_confirm\_view(request, uidb64, token):

    """

    Step 2 of 'Forgot Password':

    - Verify token

    - Show SetPasswordForm (requires fields named new\_password1 & new\_password2)

    """

    try:

        uid = force\_str(urlsafe\_base64\_decode(uidb64))

        user = User.objects.get(pk=uid)

        if not default\_token\_generator.check\_token(user, token):

            messages.error(request, 'This link has expired or is invalid.')

            return redirect('password\_reset')

        if request.method == "POST":

            # SetPasswordForm expects POST keys: new\_password1, new\_password2

            form = SetPasswordForm(user, request.POST)

            if form.is\_valid():

                form.save()  # updates user's password + clears auth sessions appropriately

                messages.success(request, 'Your password has been successfully reset.')

                return redirect('login')

            else:

                # show validation messages (length, match, common, numeric etc.)

                for field, errors in form.errors.items():

                    for error in errors:

                        messages.error(request, error)

        else:

            form = SetPasswordForm(user)

        return render(

            request, 'account/password\_reset\_confirm.html',

            {'form': form, 'uidb64': uidb64, 'token': token}

        )

    except (TypeError, ValueError, OverflowError, User.DoesNotExist):

        messages.error(request, 'An error occurred. Please try again later.')

        return redirect('password\_reset')

# account/utils.py

from django.core.mail import EmailMultiAlternatives

from django.template.loader import render\_to\_string

from django.utils.html import strip\_tags

from django.conf import settings

import threading

class SendEmailThread(threading.Thread):

    """

    Send email asynchronously so the request returns faster.

    """

    def \_\_init\_\_(self, email):

        self.email = email

        threading.Thread.\_\_init\_\_(self)

    def run(self):

        self.email.send()

def send\_activation\_email(recipient\_email, activation\_url):

    """

    Sends an HTML + plain-text activation email.

    Template: templates/account/activation\_email.html

    Context: activation\_url

    """

    subject = "Activate your account on " + settings.SITE\_NAME

    from\_email = 'no\_reply@demomailtrap.com'  # friendly sender (Mailtrap sandbox)

    to\_email = [recipient\_email]

    html\_content = render\_to\_string('account/activation\_email.html', {'activation\_url': activation\_url})

    text\_content = strip\_tags(html\_content)  # fallback for clients without HTML

    email = EmailMultiAlternatives(subject, text\_content, from\_email, to\_email)

    email.attach\_alternative(html\_content, 'text/html')

    SendEmailThread(email).start()

def send\_reset\_password\_email(recipient\_email, reset\_url):

    """

    Sends an HTML + plain-text password reset email.

    Template: templates/account/reset\_password\_email.html

    Context: reset\_url

    """

    subject = 'Reset Your Password on ' + settings.SITE\_NAME

    from\_email = 'no\_reply@demomailtrap.co'  # as in your code

    to\_email = [recipient\_email]

    html\_content = render\_to\_string('account/reset\_password\_email.html', {'reset\_url': reset\_url})

    text\_content = strip\_tags(html\_content)

    email = EmailMultiAlternatives(subject, text\_content, from\_email, to\_email)

    email.attach\_alternative(html\_content, "text/html")

    SendEmailThread(email).start()

# account/urls.py

from django.urls import path

from django.contrib.auth.views import LogoutView

from account.views import (

    home, login\_view, register\_view, activate\_account,

    password\_reset\_view, password\_reset\_confirm\_view

)

urlpatterns = [

    path('', home, name='home'),

    path('login/', login\_view, name='login'),

    path('register/', register\_view, name='register'),

    # Email activation

    path('activate/<str:uidb64>/<str:token>/', activate\_account, name='activate'),

    # Forgot password

    path('password\_reset/', password\_reset\_view, name='password\_reset'),

    path('password\_reset\_confirm/<uidb64>/<token>/', password\_reset\_confirm\_view, name='password\_reset\_confirm'),

    # Logout (POST in template)

    path('logout/', LogoutView.as\_view(), name='logout'),

]

# templates/account/base.html

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <!-- Tailwind CDN for quick styling (fine for demos) -->

  <script src="https://cdn.tailwindcss.com"></script>

  <title>{% block title %}{% endblock title %}</title>

</head>

<body class="flex flex-col min-h-screen">

  <!-- Navbar -->

  <header>

    <nav class="bg-gray-800 text-white">

      <div class="max-w-screen-xl flex flex-wrap items-center justify-between mx-auto p-4">

        <a href="{% url 'home' %}" class="flex items-center space-x-3 rtl:space-x-reverse">

            <span class="self-center text-2xl font-semibold whitespace-nowrap">Auth System</span>

        </a>

        <div class="hidden w-full md:block md:w-auto" id="navbar-default">

          <ul class="font-medium flex flex-row space-x-8">

            <li><a href="{% url 'home' %}" class="block py-2 px-3" aria-current="page">Home</a></li>

            {% if user.is\_authenticated %}

              <li><a href="{% url 'customer\_dashboard' %}" class="block py-2 px-3">Dashboard</a></li>

              <li>

                <!-- Use POST for logout to avoid CSRF issues -->

                <form method="post" action="{% url 'logout' %}">

                  {% csrf\_token %}

                  <button type="submit" class="px-3 py-1 text-sm text-white bg-rose-700 rounded-lg hover:bg-rose-900">Logout</button>

                </form>

              </li>

            {% else %}

              <li><a href="{% url 'register' %}" class="block py-2 px-3">Register</a></li>

              <li><a href="{% url 'login' %}" class="block py-2 px-3">Login</a></li>

            {% endif %}

          </ul>

        </div>

      </div>

    </nav>

  </header>

  <!-- Flash messages + page content -->

  <main class="flex-grow text-white bg-gray-600">

    {% if messages %}

      {% for message in messages %}

        <span class="{{message.tags}} bg-rose-600 p-2 font-bold">{{ message }}</span><br/><br/>

      {% endfor %}

    {% endif %}

    {% block content %}{% endblock content %}

  </main>

</body>

</html>

# templates/account/home.html

{% extends 'account/base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}

  <h1 class="text-center mt-5 font-bold">Home Page</h1>

{% endblock content %}

# templates/account/login.html

{% extends 'account/base.html' %}

{% block title %}Login{% endblock title %}

{% block content %}

<div class="flex items-center justify-center px-4 py-8">

  <div class="w-full max-w-md p-8 space-y-6 bg-gray-800 rounded-lg shadow-md">

    <h2 class="text-2xl font-bold text-center">Login to Your Account</h2>

    <!-- POST to same URL, CSRF protected -->

    <form action="" method="POST" class="space-y-6" novalidate>

      {% csrf\_token %}

      <!-- Email -->

      <div>

        <label for="email" class="block text-sm font-medium">Email</label>

        <input type="email" name="email" id="email"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="you@example.com" required>

      </div>

      <!-- Password -->

      <div>

        <label for="password" class="block text-sm font-medium">Password</label>

        <input type="password" name="password" id="password"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="••••••••" required>

      </div>

      <!-- Forgot Password -->

      <div class="text-right">

        <a href="{% url 'password\_reset' %}" class="text-sm text-rose-400 hover:underline">Forgot password?</a>

      </div>

      <!-- Submit -->

      <button type="submit" class="w-full px-4 py-2 text-white rounded-md bg-rose-500 hover:bg-rose-600 focus:outline-none">Sign in</button>

    </form>

    <!-- Divider -->

    <div class="relative mt-6">

      <div class="absolute inset-0 flex items-center"><div class="w-full border-t border-gray-600"></div></div>

      <div class="relative flex justify-center text-sm">

        <span class="px-2 bg-gray-800 text-gray-400">Or</span>

      </div>

    </div>

    <!-- Create Account -->

    <div class="text-center">

      <a href="{% url 'register' %}" class="text-sm text-rose-400 hover:underline">Create an account</a>

    </div>

  </div>

</div>

{% endblock content %}

# templates/account/register.html

{% extends 'account/base.html' %}

{% load static %}

{% block title %}Register{% endblock title %}

{% block content %}

<div class="flex items-center justify-center px-4 py-8">

  <div class="w-full max-w-md p-8 space-y-6 bg-gray-800 rounded-lg shadow-md">

    <h2 class="text-2xl font-bold text-center">Create an Account</h2>

    <!-- POST to named route -->

    <form action="{% url 'register' %}" method="POST" class="space-y-6" novalidate>

      {% csrf\_token %}

      <!-- Name -->

      <div>

        <label for="name" class="block text-sm font-medium">Name</label>

        <input type="text" value="{{ form.name.value|default:'' }}" name="name" id="name"

               class="block w-full mt-1 border-gray-300 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="Your Name" required>

        {% if form.name.errors %}

          <div class="text-red-500 text-sm mt-1">

            {% for error in form.name.errors %}<p>{{ error }}</p>{% endfor %}

          </div>

        {% endif %}

      </div>

      <!-- Email -->

      <div>

        <label for="email" class="block text-sm font-medium">Email</label>

        <input type="email" value="{{ form.email.value|default:'' }}" name="email" id="email"

               class="block w-full mt-1 border-gray-300 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="you@example.com" required>

        {% if form.email.errors %}

          <div class="text-red-500 text-sm mt-1">

            {% for error in form.email.errors %}<p>{{ error }}</p>{% endfor %}

          </div>

        {% endif %}

      </div>

      <!-- Password -->

      <div>

        <label for="password" class="block text-sm font-medium">Password</label>

        <input type="password" name="password" id="password"

               class="block w-full mt-1 border-gray-300 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="••••••••" required>

        {% if form.password.errors %}

          <div class="text-red-500 text-sm mt-1">

            {% for error in form.password.errors %}<p>{{ error }}</p>{% endfor %}

          </div>

        {% endif %}

      </div>

      <!-- Confirm Password -->

      <div>

        <label for="confirm\_password" class="block text-sm font-medium">Confirm Password</label>

        <input type="password" name="confirm\_password" id="confirm\_password"

               class="block w-full mt-1 border-gray-300 rounded-md shadow-sm bg-gray-700 p-2"

               placeholder="••••••••" required>

        {% if form.confirm\_password.errors %}

          <div class="text-red-500 text-sm mt-1">

            {% for error in form.confirm\_password.errors %}<p>{{ error }}</p>{% endfor %}

          </div>

        {% endif %}

      </div>

      <!-- Submit -->

      <button type="submit" class="w-full px-4 py-2 text-white bg-rose-600 rounded-md hover:bg-rose-700 focus:outline-none">Register</button>

    </form>

    <!-- Divider -->

    <div class="relative mt-6">

      <div class="absolute inset-0 flex items-center"><div class="w-full border-t border-gray-300"></div></div>

      <div class="relative flex justify-center text-sm">

        <span class="px-2 bg-gray-800 text-gray-400">Or</span>

      </div>

    </div>

    <!-- Login link -->

    <div class="text-center">

      <p class="text-sm text-gray-400">Existing user?

        <a href="{% url 'login' %}" class="text-rose-400 hover:underline">Login Now</a>

      </p>

    </div>

  </div>

</div>

{% endblock content %}

# templates/account/activation\_email.html

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>Account Activation</title>

</head>

<body style="font-family: Arial, sans-serif; background-color: #f5f5f5; color: #333;">

  <div style="max-width: 600px; margin: 20px auto; background: #ffffff; padding: 20px; border-radius: 8px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);">

    <h2 style="text-align: center; color: #333;">Activate Your Account</h2>

    <p>Thank you for signing up! To activate your account, please click the button below:</p>

    <!-- Activation URL injected by the view -->

    <p style="text-align: center; margin: 30px 0;">

      <a href="{{ activation\_url }}" style="background-color: #28a745; color: #ffffff; padding: 12px 24px; text-decoration: none; border-radius: 4px; display: inline-block;">Activate Account</a>

    </p>

    <p>If you didn’t register with us, you can safely ignore this email.</p>

    <hr style="border-top: 1px solid #e0e0e0; margin: 30px 0;">

    <!-- Social links (placeholders) -->

    <div style="text-align: center; margin-top: 20px;">

      <span>Follow us on: </span>

      <div style="margin-top: 10px;">

        <a href="#" style="margin: 5px; padding: 8px 12px; border-radius: 4px; background-color: #E1306C; color: #fff; text-decoration: none; font-size: 12px;">Instagram</a>

        <a href="#" style="margin: 5px; padding: 8px 12px; border-radius: 4px; background-color: #1da1f2; color: #fff; text-decoration: none; font-size: 12px;">Twitter</a>

        <a href="#" style="margin: 5px; padding: 8px 12px; border-radius: 4px; background-color: #0077B5; color: #fff; text-decoration: none; font-size: 12px;">LinkedIn</a>

        <a href="#" style="margin: 5px; padding: 8px 12px; border-radius: 4px; background-color: #ff0000; color: #fff; text-decoration: none; font-size: 12px;">YouTube</a>

      </div>

    </div>

    <p style="margin-top: 30px; font-size: 12px; color: #888;">

      If you have any issues activating your account, please contact us at:

      <a href="mailto:contact@geekyshows.com" style="color: #28a745;">contact@geekyshows.com</a>

    </p>

  </div>

</body>

</html>

# templates/account/password\_reset.html

{% extends 'account/base.html' %}

{% block title %}Reset Password{% endblock title %}

{% block content %}

<div class="flex items-center justify-center px-4 py-8">

  <div class="w-full max-w-md p-8 space-y-6 bg-gray-800 rounded-lg shadow-md">

    <h2 class="text-2xl font-bold text-center">Reset Your Password</h2>

    <!-- Important: action="" posts to the same view; CSRF protected -->

    <form action="#" method="POST" class="space-y-6" id="reset-password-form" novalidate>

      {% csrf\_token %}

      <!-- Email field (matches forms.PasswordResetForm) -->

      <div>

        <label for="email" class="block text-sm font-medium">Email</label>

        <input type="email" value="{{ form.email.value|default:'' }}" name="email" id="email"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 focus:border-rose-500 focus:ring-rose-500 p-2"

               placeholder="you@example.com" required>

        <!-- Inline form errors -->

        {% if form.email.errors %}

          <div class="text-red-500 text-sm mt-1">

            {% for error in form.email.errors %}<p>{{ error }}</p>{% endfor %}

          </div>

        {% endif %}

      </div>

      <!-- Submit -->

      <button type="submit" id="submit-btn"

              class="w-full px-4 py-2 text-white bg-rose-500 rounded-md hover:bg-rose-600 focus:outline-none">

        Send Password Reset Link

      </button>

    </form>

  </div>

</div>

<!-- UX: disable button on submit to prevent double-click -->

<script>

  const form = document.getElementById('reset-password-form');

  const submitBtn = document.getElementById('submit-btn');

  form.addEventListener('submit', function () {

    submitBtn.disabled = true;

    submitBtn.innerText = 'Sending...';

  });

</script>

{% endblock content %}

# templates/account/password\_reset\_confirm.html

{% extends 'account/base.html' %}

{% block title %}Password Reset Confirm{% endblock title %}

{% block content %}

<div class="flex items-center justify-center px-4 py-8">

  <div class="w-full max-w-md p-8 space-y-6 bg-gray-800 rounded-lg shadow-md">

    <h2 class="text-2xl font-bold text-center">Set New Password</h2>

    <!-- IMPORTANT: SetPasswordForm expects inputs named exactly new\_password1 and new\_password2 -->

    <form action="#" method="POST" class="space-y-6">

      {% csrf\_token %}

      <div>

        <label for="new\_password1" class="block text-sm font-medium">New Password</label>

        <input type="password" name="new\_password1" id="new\_password1"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 p-2" required>

      </div>

      <div>

        <label for="new\_password2" class="block text-sm font-medium">Confirm Password</label>

        <input type="password" name="new\_password2" id="new\_password2"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 p-2" required>

      </div>

      <button type="submit" class="w-full px-4 py-2 text-white bg-rose-500 rounded-md hover:bg-rose-600">Set Password</button>

    </form>

  </div>

</div>

{% endblock content %}

# templates/account/reset\_password\_email.html

<!DOCTYPE html>

<html>

<head><meta charset="UTF-8"></head>

<body>

  <p>You're receiving this email because you requested a password reset for your user account.</p>

  <p><i>Password Reset Link is valid for 1 hour only</i></p>

  <br/>

  <!-- Button-style link to the reset URL -->

  <p style="margin-bottom:100px">

    <a style="display: inline-block; padding: 10px 20px; background-color: red; color: white; text-decoration: none;"

       href="{{ reset\_url }}">Reset Password</a>

  </p>

  <span>Follow us: </span>

  <div style="display: flex; justify-content: center; align-items: center;">

    <a href="#" style="margin: 5px; padding: 5px ; border-radius: 10%; background-color: #E1306C; color: white; text-decoration: none; font-size:10px">Instagram</a>

    <a href="#" style="margin: 5px; padding: 5px ; border-radius: 10%; background-color: #1da1f2; color: white; text-decoration: none; font-size:10px">Twitter</a>

    <a href="#" style="margin: 5px; padding: 5px ; border-radius: 10%; background-color: #0077B5; color: white; text-decoration: none; font-size:10px">LinkedIn</a>

    <a href="#" style="margin: 5px; padding: 5px ; border-radius: 10%; background-color: #ff0000; color: white; text-decoration: none; font-size:10px">Youtube</a>

  </div>

  <i>If you face difficulties in resetting account password kindly contact us on: contact@geekyshows.com</i><br/>

  <i>If you did not request this, please ignore this email.</i>

</body>

</html>

# customer/views.py

from django.shortcuts import render, redirect

from django.contrib.auth.forms import PasswordChangeForm

from django.contrib import messages

from django.contrib.auth import logout

from django.contrib.auth.decorators import login\_required

@login\_required

def customer\_dashboard\_view(request):

    """

    Simple dashboard page for customers.

    """

    return render(request, 'customer/dashboard.html')

@login\_required

def password\_change\_view(request):

    """

    Change password for the logged-in user using PasswordChangeForm.

    IMPORTANT: the template must use input names:

      - old\_password

      - new\_password1

      - new\_password2

    """

    if request.method == "POST":

        form = PasswordChangeForm(user=request.user, data=request.POST)

        if form.is\_valid():

            form.save()  # updates password and handles session invalidation

            logout(request)  # force re-login with new password

            messages.success(request, "Password changed successfully. Please log in with your new password.")

            return redirect('login')

        # If you want to surface field errors as flash messages, uncomment:

        # else:

        #     for field, errors in form.errors.items():

        #         for error in errors:

        #             messages.error(request, error)

    else:

        form = PasswordChangeForm(user=request.user)

    return render(request, 'customer/password\_change.html', {'form': form})

# customer/urls.py

from django.urls import path

from customer.views import customer\_dashboard\_view, password\_change\_view

urlpatterns = [

    path('dashboard', customer\_dashboard\_view, name='customer\_dashboard'),

    path('password-change/', password\_change\_view, name='password\_change'),

]

# templates/customer/password\_change.html

{% extends 'account/base.html' %}

{% block title %}Change Password{% endblock title %}

{% block content %}

<div class="flex items-center justify-center px-4 py-8">

  <div class="w-full max-w-md p-8 space-y-6 bg-gray-800 rounded-lg shadow-md">

    <h2 class="text-2xl font-bold text-center">Change Password</h2>

    <!-- IMPORTANT: field names must be old\_password, new\_password1, new\_password2 -->

    <form action="{% url 'password\_change' %}" method="POST" class="space-y-6" novalidate>

      {% csrf\_token %}

      <!-- Old password -->

      <div>

        <label for="old\_password" class="block text-sm font-medium">Old Password</label>

        <input type="password" name="old\_password" id="old\_password"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 p-2" required>

        {% if form.old\_password.errors %}

          <p class="text-sm mt-1 text-red-500">

            {% for error in form.old\_password.errors %}{{ error }}{% endfor %}

          </p>

        {% endif %}

      </div>

      <!-- New password -->

      <div>

        <label for="new\_password1" class="block text-sm font-medium">New Password</label>

        <input type="password" name="new\_password1" id="new\_password1"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 p-2" required>

        {% if form.new\_password1.errors %}

          <p class="text-sm mt-1 text-red-500">

            {% for error in form.new\_password1.errors %}{{ error }}{% endfor %}

          </p>

        {% endif %}

      </div>

      <!-- Confirm new password -->

      <div>

        <label for="new\_password2" class="block text-sm font-medium">Confirm New Password</label>

        <input type="password" name="new\_password2" id="new\_password2"

               class="block w-full mt-1 border-gray-600 rounded-md shadow-sm bg-gray-700 text-gray-200 p-2" required>

        {% if form.new\_password2.errors %}

          <p class="text-sm mt-1 text-red-500">

            {% for error in form.new\_password2.errors %}{{ error }}{% endfor %}

          </p>

        {% endif %}

      </div>

      <button type="submit" class="w-full px-4 py-2 text-white bg-rose-500 rounded-md hover:bg-rose-600">

        Change Password

      </button>

    </form>

  </div>

</div>

{% endblock content %}

# templates/customer/dashboard.html

{% extends 'account/base.html' %}

{% block title %}Customer Dashboard{% endblock title %}

{% block content %}

  <h1 class="text-center mt-6 font-bold">Customer Dashboard</h1>

  <a href="{% url 'password\_change' %}" class="bg-rose-400 p-4 m-6 inline-block rounded">Change Password</a>

{% endblock content %}

# seller/views.py

from django.shortcuts import render

def seller\_dashboard\_view(request):

    """

    Simple seller dashboard page.

    """

    return render(request, 'seller/dashboard.html')

# seller/urls.py

from django.urls import path

from seller.views import seller\_dashboard\_view

urlpatterns = [

    path('dashboard', seller\_dashboard\_view, name='seller\_dashboard'),

]

# templates/seller/dashboard.html

{% extends 'account/base.html' %}

{% block title %}Seller Dashboard{% endblock title %}

{% block content %}

  <h1>Seller Dashboard</h1>

{% endblock content %}

## tiny checklist (to avoid gotchas)

* run python manage.py makemigrations && python manage.py migrate after introducing the custom user.
* create a superuser with email: python manage.py createsuperuser.
* templates use **exact** input names where required:
  + password\_reset\_confirm.html: new\_password1, new\_password2
  + customer/password\_change.html: old\_password, new\_password1, new\_password2
* Mailtrap creds/secret key should be env vars in real deployments.

