Django Authentication System — `accounts` app

This document contains a ready-to-use `accounts` app for Django providing: - Signup / login / logout - Password reset via email (optional, uses built-in Django flow) - User profile pages and profile edit - Session management notes

> Copy the files into a Django project and add `accounts` to `INSTALLED_APPS`.

---

1. `models.py`

```
from django.conf import settings
from django.db import models
from django.contrib.auth.models import User

class Profile(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    bio = models.TextField(blank=True)
    avatar = models.ImageField(upload_to='avatars/', null=True, blank=True)
    phone = models.CharField(max_length=20, blank=True)

    def __str__(self):
        return f"Profile({self.user.username})"
```

---

2. `signals.py` (auto-create profile)

```
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.conf import settings
from .models import Profile

@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_or_update_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)
    else:
        instance.profile.save()
```

Be sure to import this module in `apps.py` or `__init__.py` so signals are registered.

3. `forms.py`

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm,
PasswordResetForm, SetPasswordForm
from django.contrib.auth import get_user_model
from .models import Profile

User = get_user_model()

class SignUpForm(UserCreationForm):
    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = ("username", "email", "password1", "password2")

class LoginForm(AuthenticationForm):
    username = forms.CharField(label="Username or Email")

class ProfileForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ('avatar', 'bio', 'phone')
```

4. `views.py`

```python
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required
from django.urls import reverse_lazy
from django.views import View
from django.contrib.auth.views import PasswordResetView, PasswordResetDoneView,
PasswordResetConfirmView, PasswordResetCompleteView

from .forms import SignUpForm, LoginForm, ProfileForm
from .models import Profile
from django.contrib.auth import get_user_model

User = get_user_model()

class SignUpView(View):
    def get(self, request):
        form = SignUpForm()
        return render(request, 'accounts/signup.html', { 'form': form })

    def post(self, request):
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('accounts:profile', username=user.username)
        return render(request, 'accounts/signup.html', { 'form': form })

class LoginView(View):
    def get(self, request):
        form = LoginForm()
        return render(request, 'accounts/login.html', { 'form': form })

    def post(self, request):
        form = LoginForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect(request.GET.get('next') or 'home')
        return render(request, 'accounts/login.html', { 'form': form })

@login_required
def logout_view(request):
    logout(request)
    return redirect('home')

@login_required
def edit_profile(request):
    profile = request.user.profile
    if request.method == 'POST':
        form = ProfileForm(request.POST, request.FILES, instance=profile)
        if form.is_valid():
            form.save()
            return redirect('accounts:profile', username=request.user.username)
    else:
        form = ProfileForm(instance=profile)
    return render(request, 'accounts/edit_profile.html', {'form': form})

def view_profile(request, username):
    user = get_object_or_404(User, username=username)
    return render(request, 'accounts/profile.html', {'profile_user': user})

# Password reset views use Django's built-ins — use templates below
class MyPasswordResetView(PasswordResetView):
    template_name = 'accounts/password_reset_form.html'
    success_url = reverse_lazy('accounts:password_reset_done')

class MyPasswordResetDoneView(PasswordResetDoneView):
    template_name = 'accounts/password_reset_done.html'

class MyPasswordResetConfirmView(PasswordResetConfirmView):
    template_name = 'accounts/password_reset_confirm.html'
    success_url = reverse_lazy('accounts:password_reset_complete')

class MyPasswordResetCompleteView(PasswordResetCompleteView):
    template_name = 'accounts/password_reset_complete.html'
```

5. `urls.py` (inside `accounts` app)

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

app_name = 'accounts'

urlpatterns = [
    path('signup/', views.SignUpView.as_view(), name='signup'),
    path('login/', views.LoginView.as_view(), name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('profile/<str:username>/', views.view_profile, name='profile'),
    path('profile/<str:username>/edit/', views.edit_profile,
name='edit_profile'),

    # Password reset
    path('password_reset/', views.MyPasswordResetView.as_view(),
name='password_reset'),
    path('password_reset/done/', views.MyPasswordResetDoneView.as_view(),
name='password_reset_done'),
    path('reset/<uidb64>/<token>/', views.MyPasswordResetConfirmView.as_view(),
name='password_reset_confirm'),
    path('reset/done/', views.MyPasswordResetCompleteView.as_view(),
name='password_reset_complete'),
]
```

Don't forget to include `accounts.urls` in the project's main `urls.py`.

6. Templates (summary — create these under `templates/accounts/`)

- `signup.html` — registration form
- `login.html` — login form (include `{{ form.non_field_errors }}` and `{{ form.username }}`, `{{ form.password }}`)
- `profile.html` — show user info: `{{ profile_user.username }}`, `{{ profile_user.email }}`, `{{ profile_user.profile.bio }}` etc.
- `edit_profile.html` — profile form
- Password reset templates:
- `password_reset_form.html`
- `password_reset_done.html`
- `password_reset_confirm.html`
- `password_reset_complete.html`

For password-reset email body, create `registration/password_reset_email.html` (or plain text `registration/password_reset_email.txt`) containing the reset link template that Django will render.

---

7. `apps.py` (ensure signals imported)

```
from django.apps import AppConfig

class AccountsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'accounts'

    def ready(self):
        import accounts.signals
```

---

8. `settings.py` snippets

```
# Add to INSTALLED_APPS
INSTALLED_APPS += [
    'accounts',
]

# Where uploaded avatars go
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# For email password reset (development)
```

```
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'  # prints to
console

# For production use an SMTP backend, example:
# EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
# EMAIL_HOST = 'smtp.gmail.com'
# EMAIL_PORT = 587
# EMAIL_USE_TLS = True
# EMAIL_HOST_USER = 'your-email@example.com'
# EMAIL_HOST_PASSWORD = 'your-app-password'

# Login redirect & URLs
LOGIN_URL = 'accounts:login'
LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'
```

---

9. Session management notes

- Django sessions are enabled by default (`django.contrib.sessions`) — you can configure `SESSION_COOKIE_AGE` for expiry (seconds) and `SESSION_EXPIRE_AT_BROWSER_CLOSE`.
- Use `login(request, user)` to create a session; `logout(request)` will flush the session.
- To force session invalidation after password change, call `update_session_auth_hash(request, user)` when necessary.

---

10. Security & best practices

- Always validate and clean user-supplied data via forms.
- Use `EMAIL_BACKEND` appropriate for environment (console for dev, SMTP or transactional provider for prod).
- Store `EMAIL_HOST_PASSWORD` in environment variables, not in source control.
- Limit file upload sizes and validate avatar content-type.
- Consider using Django's built-in `PasswordChangeView` for logged-in users who want to change password.

---

11. Quick install steps

1. Add `accounts` to `INSTALLED_APPS`.
2. Run `python manage.py makemigrations accounts` and `migrate`.
3. Configure `MEDIA_ROOT` and `urls.py` to serve media in dev:

```python
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # ...
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

1. Create templates and wire the URLs into your site's navigation.

If you'd like, I can: - generate ready-to-copy template HTML files, - adapt the code to use a custom user model (AbstractUser) instead of default User, - add email provider configuration examples (SendGrid / Gmail), or - scaffold unit tests for these views.

Tell me what you'd like next and I'll update the app.