```python
import secrets
import os
from flask import Flask,flash,redirect,render_template,url_for,request,abort
from flask_blog.forms import RegistrationForm,LoginForm,UpdateAccountForm,PostForm
from flask_blog.models import User,Post
from flask_blog import app,db,bcrypt
from PIL import Image
from flask_login import login_user,current_user,logout_user,login_required



@app.route("/")
@app.route('/home')
def home():
    page = request.args.get('page',1,type=int)
    posts = Post.query.order_by(Post.date_posted.desc()).paginate(page = page, per_page = 5)
    return render_template('home.html',posts = posts)


@app.route("/user/<string:username>")
def user_posts(username):
    page = request.args.get('page',1,type=int)
    user = User.query.filter_by(username=username).first_or_404()
    posts = Post.query.filter_by(author=user)\
        .order_by(Post.date_posted.desc())\
        .paginate(page = page, per_page = 5)
    return render_template('user_posts.html',posts = posts,user = user)



@app.route("/about")
def about():
     return render_template('about.html',title ="About")


@app.route("/register",methods = ['GET','POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username =form.username.data, email=form.email.data, password =hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created! You are now able to login','success')
        return redirect(url_for('login'))
    return render_template('register.html',title ="Register",form = form)


@app.route("/login",methods = ['GET','POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
```

```
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email = form.email.data).first()
        if user and bcrypt.check_password_hash(user.password,form.password.data):
            login_user(user,remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash('Login Unsuccessful, Please check email and password!','danger')
    return render_template('login.html',title ="Login",form = form)


@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('home'))

def save_picture(form_picture):
    random_hex = secrets.token_hex(8)
    _,f_ext = os.path.splitext(form_picture.filename)
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(app.root_path,'static/profile_pics/',picture_fn)
    # form_picture.save(picture_path)

    i = Image.open(form_picture)
    output_size = (125,125)
    i.thumbnail(output_size)
    i.save(picture_path)

    return picture_fn

@app.route("/account", methods = ['GET','POST'])
@login_required
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash("Your account was updated successfully!",'success')
        return redirect (url_for('account'))

    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
    return render_template('account.html',title="Account",image_file=image_file,form= form)


@app.route("/post/new-post", methods = ['GET','POST'])
def new_post():
```

```
    form = PostForm()
    if form.validate_on_submit():
        post = Post(title =form.title.data,content = form.content.data, author = current_user)
        db.session.add(post)
        db.session.commit()
        flash('Your post has been submitted!','success')
        return redirect(url_for('home'))
    return render_template('create_post.html', title = 'New Post',form=form,legend = 'Create Post')


@app.route('/post/<int:post_id>')
def post(post_id):
    post = Post.query.get_or_404(post_id)
    return render_template('post.html', title = post.title, post=post)


@app.route('/post/<int:post_id>/update',methods = ['GET','POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author.username != current_user.username:
        abort(403)
    form = PostForm()
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        flash('Your post has been updated','success')
        return redirect(url_for('post',post_id = post.id))

    if request.method == "GET":
        form.title.data = post.title
        form.content.data = post.content
    return render_template('create_post.html', title = 'Update Post',form=form,legend = 'Update Post')



@app.route('/post/<int:post_id>/delete',methods = ['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(404)
    db.session.delete(post)
    db.session.commit()
    flash('Your post has been deleted!','success')
    return redirect(url_for('home'))
```