07 – User Account and Profile Picture

ရှေ့သင်ခန်းစာမှာ ဖန်တီးခဲ့တဲ့ Account template ကို လိုချင်တဲ့ profile ပုံစံဖြစ်အောင် ပြန်ပြင်ပါမယ်။ ယခု ဖန်တီးမယ့်ပုံစံမှာ username အပြင် email address နဲ့ profile picture ထပ်ထည့်ပါမယ်။ အဲဒီအတွက် account template မှာ code snippet ထည့်ပြီး လိုအပ်တာလေးတွေပြင်ပါမယ်။ code ထည့်ပြီး လိုတာ ပြင်ကြည့်ရအောင်။

```
<div class="media">
    <img class="rounded-circle account-img" src="userimage.jpg">
    <div class="media-body">
      <h2 class="account-heading">Username</h2>
      <p class="text-secondary">username@gmail.com</p>
    </div>
  </div>
```

Username နဲ့ email နေရာမှာ current_user ရဲ့ အမည်နဲ့ email လိပ်စာ အစားထိုးပေးဖို့ variable block နဲ့ ပြောင်းပေးပါမယ်။

```
Username -> {{ current_user.username }}
username@gmail.com -> {{ current_user.email }}
userimage.jpg -> {{ image_file }}
```

userimage.jpg က profile picture အနေနဲ့ထားမဲ့ image file name ဖြစ်ပါတယ်။ user တွေက မိမိရဲ့ profile ကို အမျိုးမျိုးရွေးချယ်ပြောင်းလဲလိုက်တဲ့ ပုံတွေ အပါအဝင်၊ default သတ်မှတ်ထားတဲ့ ပုံ files တွေသိမ်းဖို့ folder တစ်ခုကို static folder အောက်မှာ ဖန်တီးပေးထားရပါမယ်။ ပြီးရင် current_user က အသုံးပြုမဲ့ image file ကို account route မှာ assign လုပ်ပြီး render လုပ်ဖို့ argument အနေနဲ့ ထည့်ပေးလိုက်ရပါမယ်။

```
@app.route("/account")
@login_required
def account():
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
```

```
    return render_template('account.html', title='Account', image_file=image_file)
```

file ကို run ကြည့်ပါ။ profile picture အပါအဝင် username ၊ email လိပ်စာကို ပြထားပါလိမ့်မယ်။ အကယ်၍ username၊ email နဲ့ profile ပုံတွေကို ပြင်ချင်ရင်တော့ အချက်အလက် ရွေးချယ်ထဲ့ပေးဖို့ Update form တစ်ခုလိုပါမယ်။ UpdateAccountForm ဖန်တီးကြည့်ပါမယ်။ ပထမ username နဲ့ email ကို ပြင်ဖို့form.py file ကိုဖွင့်ပြီး code ကိုရေးပါ။

```
class UpdateAccountForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2,
                                                                           max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Update')

    def validate_username(self, username):
    if username.data != current_user.username:
        user = User.query.filter_by(username=username.data).first()
        if user:
        raise ValidationError('That username is taken. Please choose a different
                                                                    one.')

    def validate_email(self, email):
    if email.data != current_user.email:
        user = User.query.filter_by(email=email.data).first()
        if user:
        raise ValidationError('That email is taken. Please choose a different one.')
```

username နဲ့ email ကို validate လုပ်တဲ့အခါ current_user ဟုတ်မဟုတ် စစ်ပေးဖို့လိုတဲ့အတွက် validate_username() method နဲ့ validate_email() တို့ကို ပြင်ပေးရပါအုံးမယ်။ ပြီးရင်တော့ account template မှာ <form> elements ကို ထည့်ပေးပါ။ နောက် route မှာ form ကို render လုပ်ပြီး ပြန် run ကြည့်ပါ။ username နဲ့ email ကို ပြင်ဖို့ form အဆင်ပြေသွားပါမယ်။

```
@app.route("/account")
@login_required
def account():
form = UpdateAccountForm()
        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)
```

username ၊ email ပြောင်းဖို့ update button ကို နှိပ်လိုက်ရင်၊ validate_on_submit() function က သက်ဆိုင်ရာ request method အရ အလုပ်လုပ်ပေးပါလိမ့်မယ်။ အဲဒီအတွက် code ရေးပါမယ်။

```
@app.route("/account", methods=['GET', 'POST')
@login_required
def account():
form = UpdateAccountForm()
        if form.validate_on_submit():
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated!', '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)
```

run လို့ အဆင်ပြေသွားရင်တော့ ပုံပြင်ဖို့အတွက် form.py ကို ပြန်သွားပါမယ်။ picture field ကို Flask-WTF ရဲ့ Filefield အဖြစ်ဖန်တီးပြီး FileAllowed validator နဲ့ ခွင့်ပြုမဲ့ file extension ကို ကန့်သတ်ပါမယ်။ လိုအပ်တာကို import လုပ်ထားပါမယ်။ ပြီးရင် picture field ကိုထည့်ပါ။

```
picture = FileField('Update Profile Picture', validators=[FileAllowed(['jpg', 'png'])])
```

account template မှာ File field သွားထည့်ပါမယ်။

```
            <div class="form-group">
            {{ form.picture.label() }}
        {{ form.picture(class="form-control-file") }}
        {% if form.picture.errors %}
            {% for error in form.picture.errors %}
            <span class="text-danger">{{ error }}</span></br>
              {% endfor %}
           {% endif %}
        </div>
```

File ကို run ပြီး file field မှ file ကို ရွေးကြည့်ပါ။ .jpg နဲ့ .png extension နဲ့ မဟုတ်တဲ့ file ကို ရွေးရင် error ပြပါလိမ့်မယ်။ နောက် user ရွေးချယ်လိုက်မဲ့ profile picture ကို သိမ်းဖို့ သီးခြား function တစ်ခုကို route file မှာ account route ရဲ့ အပေါ်မှာ သွားရေးပါမယ်။ picture file ကို secret token နဲ့ သိမ်းမှာဖြစ်ပြီး၊ file သိမ်းမယ့် လမ်းကြောင်းကို ပြောပေးရမှာမို့ လိုအပ်တဲ့ secrets နဲ့ os ကို import လုပ်ရပါမယ်။ ပြီးရင်တော့ function code ကို အောက်ပါအတိုင်း ရေးပေးပါ။

```
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)
```

အခု ရွေးချယ်လိုက်တဲ့ profile picture များဟာ file size ကြီးနေတတ်ပါတယ်။ file size အရွယ်အစား ကြီးနေခြင်းက application ကို လေးလံနှေးကွေးစေပါတယ်။ ဒါကြောင့် picture ရဲ့ size ကို ပြင်ပါမယ်။ အဲဒီအတွက် pillow module ကို install လုပ်ပြီး Image class ကို import လုပ်ပါ။ save_picture function မှာ code ကို ဖြည့်ရေးပြီး accout route မှာ လိုတာထပ်ပြင်ပေးပါ။

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

ပြီးရင် profile picture ပြောင်းကြည့်ပြီး file သိမ်းထားသော profile_pics folder မှာ file name၊ file size များကို စစ်ကြည့်ပါ။

forms.py

```
from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileAllowed
from flask_login import current_user
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, Email, EqualTo, ValidationError
from flaskblog.models import User

class RegistrationForm(FlaskForm):
        ...

class LoginForm(FlaskForm):
        ...




class UpdateAccountForm(FlaskForm):
        username = StringField('Username', validators=[DataRequired(), Length(min=2,
max=20)])
        email = StringField('Email', validators=[DataRequired(), Email()])
        picture = FileField('Update Profile Picture', validators=[FileAllowed(['jpg', 'png'])])
        submit = SubmitField('Update')
        def validate_username(self, username):
        if username.data != current_user.username:
                user = User.query.filter_by(username=username.data).first()
```

```
            if user:
            raise ValidationError('That username is taken. Please choose a different
one.')

    def validate_email(self, email):
    if email.data != current_user.email:
            user = User.query.filter_by(email=email.data).first()
            if user:
            raise ValidationError('That email is taken. Please choose a different one.')
```

routes.py

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

posts = [
    ...
]

@app.route("/")
```

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

@app.route("/register", methods=['GET', 'POST'])
def register():
...
    return render_template('register.html', title='Register', form=form)



@app.route("/login", methods=['GET', 'POST'])
def login():
    ...
    return render_template('login.html', title='Login', form=form)

@app.route("/logout")
def logout():
...

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)
```

```
    output_size = (125, 125)
    i = Image.open(form_picture)
    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 has been updated!', '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)
```

account.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
      <div class="media">
        <img class="rounded-circle account-img" src="{{ image_file }}">
        <div class="media-body">
          <h2 class="account-heading">{{ current_user.username }}</h2>
          <p class="text-secondary">{{ current_user.email }}</p>
        </div>
      </div>
        <form method="POST" action="" enctype="multipart/form-data">
            {{ form.hidden_tag() }}

            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Account Info</legend>
                <div class="form-group">
                    {{ form.username.label(class="form-control-label") }}
                    {% if form.username.errors %}
                        {{ form.username(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.username.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
```

```
            {{ form.username(class="form-control form-control-lg") }}
        {% endif %}
    </div>
    <div class="form-group">
        {{ form.email.label(class="form-control-label") }}
        {% if form.email.errors %}
            {{ form.email(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
                {% for error in form.email.errors %}
                    <span>{{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.email(class="form-control form-control-lg") }}
        {% endif %}
    </div>
    <div class="form-group">
        {{ form.picture.label() }}
        {{ form.picture(class="form-control-file") }}
        {% if form.picture.errors %}
            {% for error in form.picture.errors %}
                <span class="text-danger">{{ error }}</span></br>
            {% endfor %}
        {% endif %}
    </div>
</fieldset>
<div class="form-group">
```

```
            {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
</div>
{% endblock content %}
```

Resources

---

python -m pip install --upgrade pip

pip install pillow