העלאת קבצים לשרת-FLASK

העלאת תמונות

בנתיים לא עדכנו את המבנה שיאפשר העלאה של תמונות לשרת, כדי לאפשר זאת צריך להשתמש בשדה מיוחד של wtforms שנקרא FileField וכדי לוודא שהקובץ שמועלה מסוג תמונה נצטרך גם איזשהו אובייקט שבודק את זה וגם אותו wtforms מספקת והוא נקרא FileAllowed. את שני האובייקטים נייבא מ-flask_wtf.file:

```
from flask_wtf.file import FileField, FileAllowed
```

נלך לטופס UpdateAccountForm ונוסיף שדה חדש מעל לשדה ה-submit, הוא צריך להיות מסוג FileField שהארגומנט הראשון שלו הוא שם השדה, והשני הוא אילו validators יש לו, כרגע ה-validator היחיד שלו זה שיהיה FileAllowed לסוג מסויים של קובץ, משום שזאת תמונה נאפשר רק קבצים מסוג ipg או png :

```
class UpdateAccountForm(FlaskForm):
    ...
    picture = FileField('Update Profile Picture', validators=[FileAllowed(['jpg', 'png'])])
    submit = SubmitField('Update')
    ...
```

נעבור לדף ה-html של account. כדי להוסיף את החלק של התמונה לתצוגת האתר נצטרך לעשות משהו דומה למה שעשינו בשני השדות הקודמים , רק לשנות כמה נקודות קטנות.

תחילה נוסיף div חדש מעל ל-div של כפתור ה-submit. העיצוב עדיין יהיה form-group של bootstrap. גם הפעם נעשה בדיקה אם היו שגיאות במהלך טעינת הקוד, אבל הפעם נצטרך להוסיף תגית span class='text-danger'> לפני החריגה כי לאובייקט FileField אין מצב של חריגה (אין פידבק). עוד משהו שצריך להוסיף הוא: כשמכריזים על הטופס (בערך שורה 11) ומגדירים method ו-action צריך גם להגדיר משתנה enctype שמגדיר איך ההודעה אמורה להתקודד בשעה שהיא עוברת לשרת. אנחנו נגדיר אותו multipart/form-data שהוא עבור העלאת קבצים.זהו שדה חובה אם רוצים לעלות קבצים עם flask:

```
...
<form method="POST" action="" enctype="multipart/form-data">
...
<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>
    ...
```

אם נריץ אמור להופיע כפתור להעלאת קבצים בדף account של משתמש פעיל. כרגע אין איזושהי לוגיקה אז הוא לא שומר את האובייקט שהעלנו, אבל הוא כן בודק שהעלנו קובץ תקין.

אז בואו נוסיף לוגיקה לכפתור. נחזור ל-routes.py ונוסיף פונקציה חדשה שמקבלת את התמונה ושמורת אותה בתיקייה profile_pics עם שם ייחודי. נרצה להגדיר את כל התמונות של המשתמשים בשם שונה, שלא בטעות נכניס למשתמש מסויים תמונה שלא שלו. לשם כך נשתמש בפונקציה שלמדנו בפרק הקודם של הספרייה secrets . כמו כן נצטרך להשיג את סוג התמונה כדי לשרשר את שמה החדש לסוג, ולבסוף לשמור אותה בתיקייה המתאימה.

בשביל להשיג את סוג התמונה נוכל להתמש בפונקציה splitetext של הספרייה os במרחב השם path.
הפונקציה מקבלת שם של קובץ ומחזירה tuple עם שם האובייקט והסוג שלו (=הסיומת של הקובץ). כדי לשמור את התמונה נצטרך לדאוג שיהיה לה מיקום מתאים, נשתמש בפונקציה join() של os.path כדי לצרף את התמונה לתיקייה profile_pics ובסוף נשתמש בפונקציה save() של אובייקטים מוסג file שמקבלת path לקובץ ושומרת אותו.

דחיסת תמונות

בעיה שעלולה לעלות בשמירת התמונה הוא שהתמונה יכולה להיות גדולה וכבדה מאוד, מה שיכול להאט את האתר.
פתרון אפשרי יהיה להשתמש בספרייה Pillow. הספרייה Pillow (עם P גדולה) היא ספרייה לעיבוד תמונות. לא ניכנס לפרטים מה בדיוק אפשר לעשות איתה, מה שחשוב לנו כרגע הוא שניתן לכווץ גודל של תמונות דרכה.
הספרייה חיצונית, לכן נצטרך להוריד אותה קודם:

```
pip install Pillow
```

לאחר ההתקנה נצטרך לייבא את הספרייה שנקראת PIL כשמייבאים אותה, ואת המחלקה Image שלה.
להגדיר גודל ולהשתמש במתודה thumbnail() שמקבלת את הגודל ומחזירה את התמונה בגודל החדש.
לאחר שכיווצנו את התמונה נשמור אותה ונחזיר אותה (את השם שלה) מהפונקציה כדי שנוכל לשמור אותה במסד נתונים:

```
import os
import secrets
from PIL import Image
...
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)
    img_file = Image.open(form_picture)
    img_file.thumbnail(output_size)
    img_file.save(picture_path)

     return picture_fn
```

ובפונקציית הניתוב נברר האם נשלחה תמונה, כלומר האם השדה form.picture.data לא ריק, ואם הוא לא נפעיל את הפונקציה save_picture עם התמונה שקיבלנו ונדאג שהמשתמש יקרא לתמונה החדשה שלו במקום הישנה:

```
...
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.profile_img = picture_file
        current_user.username = form.username.data
...
```