FLASK-FILES UPLOADING

בפרק הקודם ראינו כיצד לבצע פונקציית hashing על סיסמאות, איך לשמור אותן במסד ולאמת שאכן הוכנסה הסיסמא והמייל הנכונים.
ראינו איך יוצרים validationError מתואם אישית, ואיך להחתחבר ולהתנתק בפועל למערכת או מהמערכת.
בסוף הפרק התחלנו לבנות עמוד account שמאפשר למשתמש לשנות פרטים אישיים.
בפרק זה נמשיך לממש את העמוד- נראה איך אפשר לשנות נתונים מהמסד, ואיך לעדכן את תמונת הפרופיל של המשתמש.

טופס UpdateAccountForm-

אם אנחנו רוצים לשנות את הפרטים של המשתמש, צריך לבנות איזשהו טופס שעליו נבצע את השינויים.
נכנס ל-forms.py ונוסיף טופס חדש שישמש לעדכון החשבון.
נשים לב שהטופס יהיה יחסית זהה לטופס RegiserationForm למעט אולי הסיסמא ואימות הסיסמא, אבל זה תלוי אם נרצה לשנות את זה בטופס או במייל, כרגע נשאיר בלי האפשרות לשנות סיסמא.
עכשיו אם המשתמש לא ירצה לשנות את שם המשתמש שלו אלא רק את המייל, או להפך, הרי שבכל מקרה צריך לבצע בדיקה שהשם או המייל שהוא הכניס לא שמור כבר במסד, ואם הוא לא שינה את שם המשתמש שלו המערכת תזהה את זה כניסיון להשתמש בשם משתמש תפוס כי היא תקבל את השם המקורי שלו ממתודת ה-post של הטופס.
בשביל זה נצטרך לוודא שהמשתמש שמחובר כרגע הכניס פרטים ששונים מהפרטים הנוכחים שלו, אחרת לא נבצע בדיקה (כי הוא הכניס את השם של עצמו). כדי לדעת פרטים על המשתמש שמחובר כעת נשתמש ב-current_user של flask_login, אז לא לשכוח לייבא אותו:

```
from flask_login import current_user
...
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.')
```

עכשיו נרצה לעשות כמה שינויים קוסמטיים.
דבר ראשון נרצה להוסיף תיקייה חדשה, בתוך static שתכיל תמונת פרופיל, נקרא לה profile_pics.
לקחנו איזושהי תמונה שמצאנו באינטרנט שתייצג תמונת פרופיל דיפולטיבית.

עדכון השם והמייל של המשתמש-

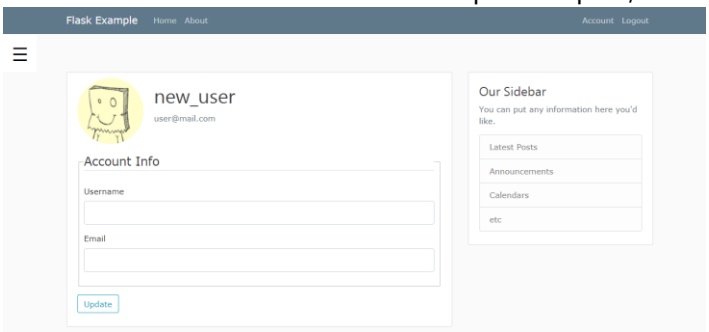
שינוי קוסמטי נוסף יהיה להוסיף את התמונת פרופיל לעמוד account, כאן נשתמש בקטע קוד html מוכן מראש כדי שיתאים את התמונה בצורה יפה לראש העמוד, נלך ל-account.html ונשנה אותו לזה:

```
{% 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>
     <!--CONTENT-->
   </div>
```

בנוסף נרצה להציג את מבנה הטופס, לכן נעתיק את המבנה (ה-html) של דף הרישום בחלקים של הטופס, נדביק במקום <--CONTENT--!> ונמחק את השדות שלא רלוונטים אלינו (שהם הסיסמא ואימות הסיסמא).
כפי שחלקכם שמו לב אנחנו משתמשים במשתנה שנקרא image_file, אי לכך נצטרך לשלוח אותו בפונקציית הניתוב.
אז נעבור ל-routes.py, נייבא את המחלקה UpdateAcountForm ונלך לפונקציה account() .
דבר ראשון ניצור משתנה עבור תמונת הפרופיל. כרגע התמונה היחידה שיש לנו היא התמונה הדיפולטיבית, ועדיין אין אפשרות לשנות אותה. נשתמש בפונקציה url_for כדי לקבל את כתובת התמונה. לפונקציה נכניס את 'static' כפרמטר ראשון והפרמטר השני יהיה התיקייה של התמונה שרשור עם שם התמונה של המשתמש, בהמשך נשמור את כל התמונות של המשתמשים בתיקייה הזאת. ונעביר את התמונה כפרמטר לפונקציית הרינדור render_template().
וכמובן ניצור אינסטנס חדש לאובייקט UpdateAccountForm ונעביר אותו לפונקציית הרינדור גם כפרמטר:

```
...
from flask_example.forms import ... , UpdateAccountForm
...
@app.route("/account")
@login_required
def account():
    form = UpdateAccountForm()
    image_file = url_for('static' , filename= f'profile_pics/{current_user.profile_img}')
    return render_template('account.html', title='Account', image_file = image_file , form = form)
```

נשמור, נריץ ואם אין משתמש מחובר ניכנס לאיזשהו משתמש שהגדרנו. אנחנו אמורים לקבל משהו כזה בערך:



שני דברים שצריך להוסיף: 1. שהשינויים ישמרו במסד הנתונים. 2.שנוכל להטעין תמונה.
אבל לפני זה, משהו נחמד שכדאי להוסיף זה שמתי שהמשתמש נכנס למערכת כדאי שבתיבת הטקסט ב-account יופיעו

השם והמייל הנוכחים שלו.
אז דבר ראשון שחשוב שנוסיף לפונקציה זה מתודות GET ו-SET. אח"כ נבצע בדיקה האם השליחה של ההודעה הייתה תקנית עם הפונקציה validate_on_submit(), אם כן נשמור את השינויים במסד נתונים.
אחד היתרונות הגדולים של sqlalchemy זה הפשטות של ביצוע פעולות שכיחות על מסדי נתונים.
אם נרצה לשנות את השם של המשתמש או המייל שלו, פשוט נשנה את השדה המתאים ב-current_user ונבצע commit() בסוף. למשל כדי לשנות את השם של המשתמש: current_user.username = form.username.data
אחרי שנשנה את שם המשתמש והסיסמא שלו נשלח גם הודעת flash שמודיעה שהשינויים בוצעו כהלכה.
אמרנו גם שיהיה נחמד לו התיבות טקסט היו מגיעות עם שם המשתמש והמייל הנוכחים כשנכנסים לעמוד account, כדי לעשות את זה נשים לב לדבר הבא: כשאנחנו מבקשים מהשרת לעבור לדף מסויים אנחנו מבצעים מתודת get, ומתי ששולחים הודעה משתמשים במתודת post. נרצה לבדוק האם ההודעה שקראה לפונקציה היא מסוג get ואם כן לאכלס את התיבות טקסט של הדף עם הנתונים הנוכחיים של המשתמש. כדי לעשות את זה נצטרך רק לשים ערכים למשתנים form.username.data ו- form.email.data ולתת להם את הערכים של current_user.username ו- current_user.email בהתאמה והם יופיעו אוטומטית בעמוד:

```
...
@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= f'profile_pics/{current_user.profile_img}')
    return render_template('account.html', title='Account', image_file = image_file , form = form)
```

נשמור ,נריץ ואנחנו אמורים לראות את שמות השדות של המשתמש מופעים כבר התיבות הטקסט של דף account, ואם ננסה לשנות את שם המשתמש או המייל הפעולה אמורה לעבוד כמו שצריך, ואפילו אנחנו מקבלים הודעה אינפורמטיבית שהחשבון עודכן.

העלאת תמונות לשרת-

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-grop של bootstrap. גם הפעם נעשה בדיקה אם היו שגיאות במהלך טעינת הקוד, אבל הפעם נצטרך להוסיף תגית לפני החריגה כי לאובייקט FileField אין מצב של חריגה(אין פידבק).
עוד משהו שצריך להוסיף הוא מתי שמכריזים 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.

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 שהמתשמש 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
...
```