from flask import Flask, render\_template, request, session, escape,\

                    redirect, url\_for, flash, g, send\_from\_directory, abort

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from werkzeug.utils import secure\_filename

import os

import urllib.parse, hashlib

ALLOWED\_EXTENSIONS = set(["png", "jpg", "jpge", "gif", "pdf"])

app = Flask(\_\_name\_\_)

app.config.from\_object("config.DevelopmentConfig")

db = SQLAlchemy(app)

def allowed\_file(filename):

    return "." in filename and filename.rsplit(".", 1)[1].lower() in ALLOWED\_EXTENSIONS

def get\_profile\_picture(email):

    default = "https://lh5.googleusercontent.com/fhcUNRdmwXqNpwf4kMHMPbn3y6eOOQnx4UfI3l0OfA308R-tI3e0cg3pFeEhxshDKyXRuZj9s8aHBqrFrmbR=w1366-h635"

    size = 512

    gravatar\_url = "https://www.gravatar.com/avatar/" + \

                        hashlib.md5(email.encode("utf-8").lower()).hexdigest() + "?"

    gravatar\_url += urllib.parse.urlencode({"d":default, "s":str(size)})

    return gravatar\_url

class Base(db.Model):

    \_\_abstract\_\_ = True

    id = db.Column(db.Integer, primary\_key=True)

    date\_created = db.Column(db.DateTime, default=db.func.current\_timestamp())

    date\_modified = db.Column(db.DateTime, default=db.func.current\_timestamp(),\

                     onupdate=db.func.current\_timestamp())

class Users(Base):

    username = db.Column(db.String(50), unique=True, nullable=False)

    email = db.Column(db.String(50), unique=True, nullable=False)

    password = db.Column(db.String(80), nullable=False)

    about\_me = db.Column(db.String(280), default="")

    images = db.relationship("Image", backref="owner", lazy="dynamic")

class Image(Base):

    filename = db.Column(db.String(80), nullable=False)

    user\_id = db.Column(db.Integer, db.ForeignKey("users.id"), nullable=False)

    owner\_username = db.Column(db.String(50), nullable=False)

@app.before\_request

def before\_request():

    if "username" in session:

        g.user = session["username"]

    else:

        g.user = None

@app.route("/")

def index():

    return render\_template("index.html")

@app.route("/search")

def search():

    username = request.args.get("username")

    return redirect(url\_for("profile", username=username))

@app.route("/signup", methods=["GET", "POST"])

def signup():

    if not g.user:

        if request.method == "POST":

            username = Users.query.filter\_by(username=request.form["username"]\

                        .lower()).first()

            email = Users.query.filter\_by(email=request.form["email"].lower()).first()

            if username or email:

                flash("Usernames and emails must be unique.", "alert-warning")

                return redirect(request.url)

            hashed\_pw = generate\_password\_hash(request.form["password"], method="sha256")

            new\_user = Users(username=request.form["username"].lower(), \

                            email=request.form["email"].lower(), password=hashed\_pw)

            db.session.add(new\_user)

            db.session.commit()

            flash("You've registered successfully.", "alert-success")

            return redirect(url\_for("login"))

        return render\_template("signup.html")

    flash("You're already logged in.", "alert-primary")

    return redirect(url\_for("home"))

@app.route("/login", methods=["GET", "POST"])

def login():

    if not g.user:

        if request.method == "POST":

            user = Users.query.filter\_by(username=request.form["username"]\

                        .lower()).first()

            if user and check\_password\_hash(user.password, request.form["password"]):

                session["username"] = user.username

                flash("Now you're logged in.", "alert-success")

                return redirect("home")

            flash("Your credentials are invalid, check and try again.", "alert-warning")

        return render\_template("login.html")

    flash("You are already logged in.", "alert-primary")

    return redirect(url\_for("home"))

@app.route("/profile/<username>")

def profile(username):

    user = Users.query.filter\_by(username=username).first()

    if user:

        files = user.images.order\_by(Image.date\_modified).limit(10).all()

        picture = get\_profile\_picture(user.email)

        return render\_template("profile.html", user=user, files=files, picture=picture)

    abort(404)

@app.route("/profile/edit", methods=["GET", "POST"])

def edit\_profile():

    user = Users.query.filter\_by(username=g.user).first()

    if request.method == "POST":

        user.about\_me = request.form["about"]

        db.session.commit()

        flash("Changes has been saved successfully!", "alert-success")

        return redirect(url\_for("profile", username=g.user))

    return render\_template("edit\_profile.html", user=user)

@app.route("/home", methods=["GET", "POST"])

def home():

    if g.user:

        if request.method == "POST":

            if "file" not in request.files:

                flash("No file part.", "alert-danger")

                return redirect(request.url)

            file = request.files["file"]

            if file.filename == "":

                flash("No selected file.", "alert-warning")

                return redirect(request.url)

            if file and allowed\_file(file.filename):

                filename = secure\_filename(file.filename)

                user = Users.query.filter\_by(username=g.user).first()

                file\_to\_db = Image(filename=filename, owner=user, \

                                owner\_username=user.username)

                db.session.add(file\_to\_db)

                db.session.commit()

                file.save(os.path.join(app.config["UPLOAD\_FOLDER"], filename))

                return redirect(url\_for("get\_files"))

        return render\_template("home.html")

    flash("You must be logged in.", "alert-warning")

    return redirect(url\_for("login"))

@app.route("/files")

def get\_all\_files():

    files = Image.query.order\_by(Image.date\_modified).limit(200).all()

    return render\_template("files.html", files=files)

@app.route("/myfiles")

def get\_files():

    if g.user:

        user = Users.query.filter\_by(username=g.user).first()

        if user.images.all():

            files = user.images.order\_by(Image.date\_modified).all()

            return render\_template("my\_files.html", files=files)

        files = []

        return render\_template("my\_files.html", files=files)

    flash("You must be logged in.", "alert-warning")

    return redirect(url\_for("login"))

@app.route('/uploads/<filename>')

def uploaded\_file(filename):

    return send\_from\_directory(app.config['UPLOAD\_FOLDER'], filename)

@app.route("/logout")

def logout():

    session.pop("username", None)

    flash("You're logged out.", "alert-secondary")

    return redirect(url\_for("index"))

@app.route("/about")

def about():

    return render\_template("about.html")

@app.errorhandler(404)

def page\_not\_found(error):

    return render\_template("404\_page\_not\_found.html"), 404

if \_\_name\_\_ == "\_\_main\_\_":

    db.create\_all()

    app.run()