from flask import Flask, render\_template, redirect, url\_for, flash  
from flask\_bootstrap import Bootstrap  
from flask\_ckeditor import CKEditor, request  
from datetime import date  
from werkzeug.security import generate\_password\_hash, check\_password\_hash  
from flask\_sqlalchemy import SQLAlchemy  
from sqlalchemy.orm import relationship  
from flask\_login import UserMixin, login\_user, LoginManager, login\_required, current\_user, logout\_user  
from forms import CreatePostForm, RegisterForm, LoginForm, CommentForm  
from flask\_gravatar import Gravatar  
import os # Issues with take data for db table file  
from functools import wraps  
from flask import abort # Like AY  
  
from dotenv import load\_dotenv # install python-dotenv  
  
load\_dotenv('env/.env')  
API\_KEY = os.getenv('OWM\_API\_KEY')  
  
basedir = os.path.abspath(os.path.dirname(\_\_file\_\_)) # Issues with take data for db table file  
app = Flask(\_\_name\_\_)  
app.config['SECRET\_KEY'] = os.getenv('SECRET\_KEY')  
ckeditor = CKEditor(app)  
Bootstrap(app)  
  
# #CONNECT TO DB  
app.config['SQLALCHEMY\_DATABASE\_URI'] =\  
 'sqlite:///' + os.path.join(basedir, 'blog.db') # Issues with take data for db table file  
# app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///blog.db'  
app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  
db = SQLAlchemy(app)  
login\_manager = LoginManager()  
login\_manager.init\_app(app)  
# login\_manager.login\_view = 'login'  
  
gravatar = Gravatar(app, size=100, rating='g', default='retro', force\_default=False, force\_lower=False, use\_ssl=False,  
 base\_url=None)  
  
  
@login\_manager.user\_loader  
def load\_user(user\_id):  
 return RegisterUser.query.get(int(user\_id))  
  
  
# like AY  
def admin\_only(f):  
 @wraps(f)  
 def decorate\_function(\*args, \*\*kwargs):  
 if current\_user.id is not 1:  
 return abort(403)  
 # Otherwise continue with the route function  
 return f(\*args, \*\*kwargs)  
 return decorate\_function  
  
  
# #CONFIGURE TABLES  
class RegisterUser(UserMixin, db.Model):  
 \_\_tablename\_\_ = "users"  
 id = db.Column(db.Integer, primary\_key=True)  
 email = db.Column(db.String(250), unique=True, nullable=False)  
 password = db.Column(db.String(250), nullable=False)  
 name = db.Column(db.String(250), nullable=False)  
 posts = relationship('BlogPost', back\_populates='users')  
 comments = relationship('CommentUser', back\_populates='users')  
  
  
class BlogPost(db.Model):  
 \_\_tablename\_\_ = "blog\_posts"  
 id = db.Column(db.Integer, primary\_key=True)  
 author = db.Column(db.String(250), db.ForeignKey('users.name'))  
 users = relationship('RegisterUser', back\_populates='posts')  
 title = db.Column(db.String(250), unique=True, nullable=False)  
 subtitle = db.Column(db.String(250), nullable=False)  
 date = db.Column(db.String(250), nullable=False)  
 body = db.Column(db.Text, nullable=False)  
 img\_url = db.Column(db.String(250), nullable=False)  
 comments = relationship('CommentUser', back\_populates='blog\_posts')  
  
  
class CommentUser(db.Model):  
 \_\_tablename\_\_ = "comments"  
 id = db.Column(db.Integer, primary\_key=True)  
 post\_id = db.Column(db.Integer, db.ForeignKey('blog\_posts.id'))  
 comment = db.Column(db.Text, nullable=False)  
 author = db.Column(db.String(250), db.ForeignKey('users.name'))  
 users = relationship('RegisterUser', back\_populates='comments')  
 blog\_posts = relationship('BlogPost', back\_populates='comments')  
  
  
with app.app\_context(): # Add before add a table or tablename  
 db.create\_all()  
 db.session.commit()  
  
  
@app.route('/')  
def get\_all\_posts():  
 posts = BlogPost.query.all()  
 return render\_template("index.html", all\_posts=posts)  
  
  
@app.route('/register', methods=['GET', 'POST'])  
def register():  
 form = RegisterForm()  
 if form.validate\_on\_submit():  
 password\_hash = generate\_password\_hash(form.password.data, method='pbkdf2:sha256', salt\_length=8)  
 email = form.email.data  
 user = RegisterUser.query.filter\_by(email=email).first()  
 if not user:  
 with app.app\_context():  
 db.create\_all()  
 new\_user = RegisterUser(email=email, password=password\_hash, name=form.name.data)  
 db.session.add(new\_user)  
 db.session.commit()  
 # Log and authenticate user after adding details to database  
 login\_user(new\_user)  
 return redirect(url\_for("get\_all\_posts"))  
 else:  
 flash("You've already signed up with that email, log in instead!")  
 return render\_template("login.html", form=form)  
 return render\_template("register.html", form=form)  
  
  
@app.route('/login', methods=['GET', 'POST'])  
def login():  
 form = LoginForm()  
 if form.validate\_on\_submit():  
 email = request.form.get('email')  
 password = request.form.get('password')  
 user = RegisterUser.query.filter\_by(email=email).first()  
 if user:  
 if check\_password\_hash(user.password, password):  
 login\_user(user)  
 return redirect(url\_for("get\_all\_posts"))  
 else:  
 flash("Wrong password -Try Again!")  
 else:  
 flash("That email doesn't Exist! -Try Again!")  
 return render\_template("login.html", form=form)  
 return render\_template("login.html", form=form)  
  
  
@app.route('/logout')  
@login\_required  
def logout():  
 logout\_user()  
 return redirect(url\_for('get\_all\_posts'))  
  
  
@app.route("/post/<int:post\_id>", methods=['GET', 'POST'])  
def show\_post(post\_id):  
 form\_comment = CommentForm()  
 requested\_post = BlogPost.query.get(post\_id)  
 requested\_comments = requested\_post.comments  
 if form\_comment.validate\_on\_submit():  
 if 'UserMixin' not in str(current\_user):  
 # print(current\_user.email)  
 with app.app\_context():  
 db.create\_all()  
 new\_comment = CommentUser(post\_id=post\_id, comment=request.form.get('comment'), users=current\_user)  
 db.session.add(new\_comment)  
 db.session.commit()  
 return redirect(url\_for("get\_all\_posts"))  
 else:  
 flash("You need to login or register to comment.")  
 return redirect(url\_for("login"))  
 # comment(post\_id)  
 return render\_template("post.html", post=requested\_post, form=form\_comment, comments=requested\_comments)  
  
  
@app.route("/about")  
def about():  
 return render\_template("about.html")  
  
  
@app.route("/contact")  
def contact():  
 return render\_template("contact.html")  
  
  
@app.route("/new-post", methods=['GET', 'POST'])  
@login\_required  
@admin\_only  
def add\_new\_post():  
 form = CreatePostForm()  
 if form.validate\_on\_submit():  
 with app.app\_context():  
 db.create\_all()  
 new\_post = BlogPost(  
 title=form.title.data,  
 subtitle=form.subtitle.data,  
 img\_url=form.img\_url.data,  
 author=current\_user.name, # form.author.data, # current\_user, #this when be login  
 body=form.body.data,  
 date=date.today().strftime("%B %d, %Y")  
 )  
 db.session.add(new\_post)  
 db.session.commit()  
 return redirect(url\_for("get\_all\_posts"))  
 return render\_template("make-post.html", form=form)  
  
  
@app.route("/edit-post/<int:post\_id>", methods=['GET', 'POST'])  
@login\_required  
@admin\_only  
def edit\_post(post\_id):  
 post = BlogPost.query.get(post\_id)  
 edit\_form = CreatePostForm(  
 title=post.title,  
 subtitle=post.subtitle,  
 img\_url=post.img\_url,  
 author=post.author,  
 body=post.body  
 )  
 if edit\_form.validate\_on\_submit():  
 post.title = request.form.get('title')  
 post.subtitle = request.form.get('subtitle')  
 post.img\_url = request.form.get('img\_url')  
 post.author = request.form.get('author')  
 post.body = request.form.get('body')  
 db.session.commit()  
 return redirect(url\_for("show\_post", post\_id=post\_id))  
  
 return render\_template("make-post.html", form=edit\_form)  
  
  
@app.route("/delete/<int:post\_id>")  
@login\_required  
@admin\_only  
def delete\_post(post\_id):  
 post\_to\_delete = BlogPost.query.get(post\_id)  
 db.session.delete(post\_to\_delete)  
 db.session.commit()  
 return redirect(url\_for('get\_all\_posts'))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(host='0.0.0.0', port=5000)