ارائه مستندات پروژه تحت وب

ارائه کننده: محمدحسین عبداللهی

استاد مربوطه: امیرکیوان شفیعی

درس مربوطه: مهندسی اینترنت

مقدمه: ساخت یک اپلیکیشن تحت وب با فریمورک فلسک که در این اپلیکیشن ما قرار است وبلاگی را طراحی کنیم.

اهداف پروژه: ایجاد یک محیطی برای کاربران جهت انتشار مقالات شعری خود فراهم کردن امکان خواندن، نظر دادن و به اشتراک گذاری مقالات توسط کاربران ایجاد یک رابط کاربری ساده و کاربرپسند

این اپلیکیشن دارای ویژگی‌های زیر است:

امکان ثبت نام و ورود کاربران

امکان ایجاد، ویرایش و حذف مقالات توسط کاربران

امکان خواندن و به اشتراک گذاری مقالات توسط کاربران

یک رابط کاربری ساده و کاربرپسند

نحوه استفاده:

برای استفاده از این اپلیکیشن، ابتدا باید یک حساب کاربری ایجاد کنید. برای این کار، بر روی پیوند "ثبت نام" در صفحه اصلی کلیک کنید. سپس، اطلاعات خواسته شده را وارد کنید و بر روی دکمه "ثبت نام" کلیک کنید.

پس از ثبت نام، می‌توانید به اپلیکیشن وارد شوید. برای این کار، بر روی پیوند "ورود" در صفحه اصلی کلیک کنید. سپس، نام کاربری و رمز عبور خود را وارد کنید و بر روی دکمه "ورود" کلیک کنید.

پس از ورود به اپلیکیشن، می‌توانید مقالات خود را ایجاد کنید. برای این کار، بر روی پیوند "نوشتن مقاله" در صفحه اصلی کلیک کنید. سپس، عنوان، متن خود را وارد کنید و بر روی دکمه "ذخیره" کلیک کنید.

می‌توانید مقالات خود را ویرایش کنید یا حذف کنید. برای ویرایش یک مقاله، بر روی پیوند "ویرایش" در صفحه مقاله کلیک کنید. برای حذف یک مقاله، بر روی پیوند "حذف" در صفحه مقاله کلیک کنید.

بررسی ساختار پروژه و نحوه ایجاد آن:

ما فولدری به نام blog را ایجاد کردیم که درون این فولدر، فایل‌ها و فولدرهای پروژه قرار میگیرد.

اولین فایلی که ایجاد شده run.py بوده که برنامه در این فایل اجرا میشود.

و چندین فایل پایتونی دیگر ایجاد کردیم که به‌مرور به همه آنها می‌پردازیم.

فایل run.py :

```
from blog import app


if name == 'main':
    app.run(debug=True)
```

فایل run.py نقطه شروع اپلیکیشن است. این فایل، اپلیکیشن را راه‌اندازی می‌کند و آن را در حالت توسعه اجرا می‌کند.

در حالت توسعه، خطاها به صورت خودکار نمایش داده می‌شوند و کدهای جدید بدون نیاز به راه‌اندازی مجدد اپلیکیشن بارگذاری می‌شوند.

برای راه‌اندازی اپلیکیشن در حالت توسعه، می‌توانید از دستور زیر استفاده کنید:

```
python run.py
```

این دستور، اپلیکیشن را در پورت 5000 اجرا می‌کند.

فایل __init__.py :

این فایل، اپلیکیشن فلسک را راه‌اندازی می‌کند و تنظیمات آن را پیکربندی می‌کند. همچنین، کتابخانه‌های لازم برای تعامل با پایگاه داده، رمزنگاری رمز عبور و احراز هویت کاربر را وارد می‌کند.

from flask import Flask

from flask_sqlalchemy ی import SQLAlchemy

این خطوط کتابخانه‌های فلسک و SQLAlchemy را وارد می‌کنند که برای ایجاد یک برنامه وب با پایگاه داده ضروری هستند.

from flask_bcrypt import Bcrypt

from flask_login import LoginManager

این خطوط کتابخانه‌های Bcrypt و LoginManager را وارد می‌کنند که به ترتیب برای رمزنگاری رمز عبور و احراز هویت کاربر استفاده می‌شوند.

app = Flask(name)

'app.config['SECRET_KEY'] = '3e5e4c1b79c0df2a4dbc849e19296d1d

این خطوط اپلیکیشن فلسک را راه‌اندازی می‌کنند و کلید مخفی اپلیکیشن را تنظیم می‌کنند. کلید مخفی برای تولید امضای رمزنگاری ایمن استفاده می‌شود.

'app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///../blog.db

این خط URL پایگاه داده را برای اپلیکیشن تنظیم می‌کند. در این مورد، پایگاه داده یک فایل SQLite به نام blog.db است که در همان دایرکتوری اپلیکیشن فلسک قرار دارد.

db = SQLAlchemy(app)

این خط یک نمونه از شیء SQLAlchemy را ایجاد می‌کند و آن را با اپلیکیشن فلسک مرتبط می‌کند، که تعاملات پایگاه داده را فعال می‌کند.

bcrypt = Bcrypt(app)

این خط یک نمونه از شیء Bcrypt را ایجاد می‌کند و آن را با اپلیکیشن فلسک مرتبط می‌کند، که رمزنگاری رمز عبور را با استفاده از الگوریتم bcrypt فعال می‌کند.

login_manager = LoginManager(app)

این خط یک نمونه از شیء LoginManager را ایجاد می‌کند و آن را با اپلیکیشن فلسک مرتبط می‌کند، که احراز هویت کاربر را فعال می‌کند.

'login_manager.login_view = 'login

'login_manager.login_message = 'Please login first

'login_manager.login_message_category = 'info

این خطوط شیء LoginManager را با تنظیمات دلخواه پیکربندی می‌کنند. تنظیم login_view نشان می‌دهد که کاربر در هنگام تلاش برای دسترسی به یک مسیر محدود بدون ورود به سیستم به کجا هدایت می‌شود. تنظیم login_message پیامی را برای نمایش به کاربران غیرمجاز فراهم می‌کند. تنظیم login_message_category دسته پیام را تنظیم می‌کند که تعیین می‌کند سبک پیام چگونه باشد.

from blog import routes

این خط ماژول routes را وارد می‌کند که شامل مسیرهای اپلیکیشن است. مسیرها مسیرهای URL و توابع نمایش مربوطه را که درخواست‌های ورودی را مدیریت می‌کنند، تعریف می‌کنند.

نکته: اگر این خط کد اولین خط کد برنامه باشد و در آخر نباشد ما به مشکل ایمپورت های حلقوی دچار میشویم.

فایل routes.py :

ساختار کلی

فایل routes.py مسیرهای URL و توابع مربوطه را برای یک برنامه وبلاگ تعریف می کند. از کتابخانه های Flask، Flask-Login و Flask-Oauthlib به ترتیب برای مسیریابی، احراز هویت کاربر و عملکرد OAuth استفاده می کند.

توابع مسیریابی

home(): صفحه اصلی را مدیریت می کند و فهرستی از همه پست های منتشر شده را نمایش می دهد.

detail(post_id): جزئیات یک پست خاص را بر اساس ID پست ارائه شده نمایش می دهد.

register(): ثبت نام کاربر را مدیریت می کند، داده های فرم ارسالی را اعتبارسنجی می کند و یک حساب کاربری جدید ایجاد می کند.

login(): ورود کاربر را مدیریت می کند، داده های فرم ارسالی را اعتبارسنجی می کند و کاربر را احراز هویت می کند.

logout(): خروج کاربر را مدیریت می کند، کاربر را امضا می کند و به صفحه اصلی هدایت می کند.

فرم ها

RegistrationForm: برای ثبت نام کاربر استفاده می شود، نام کاربری، آدرس ایمیل و ورودی رمز عبور را جمع آوری می کند.

LoginForm: برای ورود کاربر استفاده می شود، ورودی آدرس ایمیل و رمز عبور را جمع آوری می کند.

UpdateProfileForm: برای به روز رسانی اطلاعات نمایه کاربر استفاده می شود، معمولاً شامل نام کاربری و آدرس ایمیل می شود.

PostForm: برای ایجاد و ویرایش پست های وبلاگ استفاده می شود، عنوان، محتوا و اطلاعات نویسنده را جمع آوری می کند.

احراز هویت

current_user: امکان دسترسی به شیء کاربری که در حال حاضر وارد شده است را فراهم می کند.

(login_user(user: کاربر ارائه شده را احراز هویت می کند و آن ها را با جلسه فعلی مرتبط می کند.

logout_user(): کاربر فعلی را خارج می کند و جلسه آن ها را باطل می کند.

login_required: دکوراتور مورد استفاده برای محافظت از مسیرهایی که نیاز به احراز هویت دارند.

OAuth

oauth: شیء مشتری OAuth Flask-Oauthlib، که برای ادغام با ارائه دهندگان OAuth استفاده می شود.

home(): این تابع صفحه اصلی را مدیریت می کند. ابتدا، آن یک شیء Post را با استفاده از عبارت پرس و جو Post.query.all() ایجاد می کند. سپس، آن شیء را به یک تابع render_template() ارسال می کند که صفحه home.html را با فهرستی از پست ها ایجاد می کند.

detail(post_id): این تابع جزئیات یک پست خاص را بر اساس ID پست ارائه شده نمایش می دهد. ابتدا، آن یک شیء Post را با استفاده از عبارت پرس و جو Post.query.get_or_404(post_id) ایجاد می کند. سپس، آن شیء را به یک تابع render_template() ارسال می کند که صفحه detail.html را با جزئیات پست ایجاد می کند.

register(): این تابع ثبت نام کاربر را مدیریت می کند. ابتدا، آن یک شیء RegistrationForm را ایجاد می کند. سپس، آن شیء را اعتبارسنجی می کند. اگر فرم معتبر باشد، آن یک شیء User جدید ایجاد می کند و آن را در پایگاه داده ذخیره می کند. سپس، آن یک پیام تأیید را به کاربر نشان می دهد.

login(): این تابع ورود کاربر را مدیریت می کند. ابتدا، آن یک شیء LoginForm را ایجاد می کند. سپس، آن شیء را اعتبارسنجی می کند. اگر فرم معتبر باشد، آن یک کاربر را از پایگاه داده با استفاده از آدرس ایمیل ارائه شده جستجو می کند.

سپس، آن رمز عبور کاربر را با رمز عبور ارائه شده مقایسه می کند. اگر رمز عبور مطابقت داشته باشد، آن کاربر را وارد می کند و یک پیام تأیید را به کاربر نشان می دهد.

logout(): این تابع خروج کاربر را مدیریت می کند. ابتدا، آن کاربر فعلی را با استفاده از تابع logout_user() خارج می کند. سپس، آن یک پیام تأیید را به کاربر نشان می دهد.

:detail(post_id)

نمایش یک پست خاص را مدیریت می کند.

پست با post_id ارائه شده را با استفاده از روش Post.query.get_or_404(post_id) بازیابی می کند.

پست بازیابی شده را برای رندر کردن به قالب detail.html ارسال می کند.

:post/<int:post_id>/delete

پست مشخص شده را حذف می کند.

با بررسی اینکه آیا نویسنده پست با کاربر فعلی وارد شده مطابقت دارد، اطمینان حاصل می کند که کاربر فعلی مجاز به حذف پست است.

پست را از پایگاه داده با استفاده از روش db.session.delete(post) حذف می کند.

تغییرات را در پایگاه داده ثبت می کند.

پیام موفقیت را با استفاده از flash('post deleted', 'info') و redirect(url_for('home')) نمایش می دهد و به صفحه اصلی هدایت می کند.

:post/<int:post_id>/update

به‌روزرسانی یک پست موجود را مدیریت می کند.

با بررسی اینکه آیا نویسنده پست با کاربر فعلی وارد شده مطابقت دارد، اطمینان حاصل می کند که کاربر فعلی مجاز به به‌روزرسانی پست است.

یک فرم برای ویرایش پست با استفاده از کلاس PostForm() نمایش می دهد.

ارسال فرم را با استفاده از روش validate_on_submit() کلاس PostForm() اعتبارسنجی می کند.

عنوان و محتوای پست را با استفاده از داده های فرم به‌روز می کند.

تغییرات را در پایگاه داده ثبت می کند.

پیام موفقیت را با استفاده از flash('post updated', 'info') و redirect(url_for('detail', post_id=post.id)) نمایش می دهد و به صفحه جزئیات پست هدایت می کند.

new_post():

ایجاد یک پست جدید را مدیریت می کند.

با بررسی دکوراتور login_required اطمینان حاصل می کند که کاربر فعلی مجاز به ایجاد پست است.

یک فرم برای ایجاد یک پست جدید با استفاده از کلاس PostForm() نمایش می دهد.

ارسال فرم را با استفاده از روش validate_on_submit() کلاس PostForm() اعتبارسنجی می کند.

یک شیء Post جدید با استفاده از داده های فرم ایجاد می کند.

پست جدید را به پایگاه داده با استفاده از روش db.session.add(post) اضافه می کند.

تغییرات را در پایگاه داده ثبت می کند.

پیام موفقیت را با استفاده از flash('post created', 'info') و redirect(url_for('home')) نمایش می دهد و به صفحه اصلی هدایت می کند.

این مسیرها عملکرد اصلی برنامه وبلاگ را مدیریت می کنند، از جمله مدیریت پست ها، نمایه های کاربر و احراز هویت کاربر. آنها استفاده از Flask، Flask-Login و Flask-WTF را برای ایجاد یک برنامه وبلاگ ایمن و کاربرپسند نشان می دهند.

توضیحات بیشتر

home():

این مسیر صفحه اصلی را مدیریت می کند که فهرستی از همه پست های منتشر شده را نمایش می دهد.

این کار با بازیابی تمام پست ها از پایگاه داده با استفاده از روش Post.query.all() انجام می شود.

سپس، این پست ها به قالب home.html ارسال می شوند تا نمایش داده شوند.

detail(post_id):

این مسیر نمایش یک پست خاص را مدیریت می کند.

این کار با بازیابی پست با post_id ارائه شده از پایگاه داده با استفاده از روش Post.query.get_or_404(post_id) انجام می شود.

سپس، این پست به قالب detail.html ارسال می شود تا نمایش داده شود.

post/<int:post_id>/delete:

این مسیر پست مشخص شده را حذف می کند.

این کار با بررسی اینکه آیا نویسنده پست با کاربر فعلی وارد شده مطابقت دارد، انجام می شود.

علاوه‌بر لاگین معمولی به این پروژه لاگین با گوگل هم اضافه شده.

کد زیر برای احراز هویت Google OAuth با Flask و کتابخانه oauth نوشته شده است. در ادامه توضیحاتی درباره هر بخش آن آمده است:

پروژه Google OAuth:

```
(google = oauth.remote_app

'google',

consumer_key='510368579935-
','491va33ab3do6mkigl6g9sn2amn34md0.apps.googleusercontent.com

','consumer_secret='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

}=request_token_params

'scope': 'email',

{,

','/base_url='https://www.googleapis.com/oauth2/v1

,request_token_url=None

','access_token_method='POST

','access_token_url='https://accounts.google.com/o/oauth2/token

','authorize_url='https://accounts.google.com/o/oauth2/auth

(
```

این قسمت پروژه Google OAuth را با اطلاعات مورد نیاز تنظیم می کند، مانند consumer_key و consumer_secret.

روت ورود Google:

```
@app.route('/login/google')
def google_login():
    return google.authorize(callback=url_for('google_authorized', _external=True))
```

این قسمت یک روت برای شروع فرایند ورود به Google تعریف می کند. این روت کاربر را به صفحه احراز هویت Google OAuth هدایت می کند.

روت بازگشت از Google:

```
@app.route('/login/google/callback')
@google.authorized_handler
def google_authorized(resp):
    ...
```

این قسمت یک روت برای دریافت بازگشت از Google پس از احراز هویت کاربر تعریف می کند. دکوریتور @google.authorized_handler بررسی می کند که کاربر احراز شده است و پاسخ را به تابع google_authorized پاس می دهد.

تابع احراز شده Google:

```
if resp is None or resp.get('access_token') is None:
    پردازش رد شدن درخواست
    ...
else:
    ذخیره توکن دسترسی در سشن
```

```
session['google_token'] = (resp['access_token'], '')
```

دریافت اطلاعات کاربر با استفاده از توکن دسترسی

```
user_info = google.get('userinfo')
```

نمایش پیام با موفقیت و انتقال به صفحه اصلی

```
flash('ورود با موفقیت انجام شد: ' + user_info.data['email'])
return redirect(url_for('home'))
```

این تابع بررسی می کند که کاربر احراز شده است و دسترسی را به کاربر داده می کند. اگر کاربر رد شده است، یک پیام خطا نمایش داده می شود. اگر کاربر احراز شده است، توکن دسترسی را در سشن ذخیره می کند، اطلاعات کاربر را با استفاده از توکن دسترسی دریافت می کند، پیام با موفقیت نمایش داده می شود و به صفحه اصلی هدایت می شود.

فایل models.py :

کدهای این فایل یک مدل کاربر و یک مدل پست برای یک وب سایت وبلاگ را تعریف می کند.

مدل کاربر

id یک کلید اصلی است که هر کاربر را به طور منحصر به فرد شناسایی می کند.

username نام کاربری کاربر را ذخیره می کند.

email آدرس ایمیل کاربر را ذخیره می کند.

password رمز عبور هش شده کاربر را ذخیره می کند.

posts یک رابطه چند به یک بین کاربران و پست ها را تعریف می کند. این بدان معناست که یک کاربر می تواند چندین پست داشته باشد و هر پست متعلق به یک کاربر است. گزینه 'backref='author مشخص می کند که ویژگی author کلاس Post به شیء User مربوطه اشاره خواهد کرد.

lazy=True مشخص می کند که پست ها به صورت تنبل بارگیری می شوند، به این معنی که فقط زمانی بارگیری می شوند که واقعاً مورد نیاز باشند. این روشی کارآمدتر برای مدیریت روابط است، به خصوص وقتی یک کاربر تعداد زیادی پست داشته باشد.

مدل پست

id یک کلید اصلی است که هر پست را به طور منحصر به فرد شناسایی می کند.

title عنوان پست را ذخیره می کند.

date تاریخ و زمان ایجاد پست را ذخیره می کند.

content محتوای پست را ذخیره می کند.

user_id به ستون id جدول User اشاره می کند، که نشان می دهد کاربری که پست را ایجاد کرده است.

این کد یک کاربر جدید با نام کاربری فلان و آدرس ایمیل فلان ایجاد می کند. سپس یک پست جدید با عنوان عنوان پست، تاریخ ایجاد datetime.now() و محتوای محتوای پست ایجاد می کند. در نهایت، کاربر فعلی را بارگیری می کند و همه پست های آن کاربر را بارگیری می کند.

نکته:

posts = db.relationship('Post', backref='author', lazy=True)

این خط درون دیتابیس ستونی ایجاد نمیکند و فقط روابط را نشان میدهد.

رابطه به صورت چند به یک است یعنی یک کاربر میتواند چند پست ایجاد کند.

فایل form.py :

فرم‌های Flask را برای ثبت نام کاربر، ورود به سیستم، به‌روزرسانی پروفایل و ایجاد پست تعریف می‌کند. این فرم‌ها برای جمع‌آوری ورودی کاربر و اعتبارسنجی آن قبل از ارسال آن به منطق برنامه برای پردازش استفاده می‌شوند.

کلاس‌های فرم

این فایل حاوی چهار کلاس فرم Flask است:

RegistrationForm: این فرم برای ثبت نام کاربر استفاده می‌شود. نام کاربری، آدرس ایمیل، رمز عبور و رمز عبور تأیید را از کاربر جمع‌آوری می‌کند. همچنین داده‌ها را برای اطمینان از اینکه نام کاربری منحصربه‌فرد است، ایمیل معتبر است و گذرواژه‌ها مطابقت دارند، اعتبارسنجی می‌کند.

LoginForm: این فرم برای ورود به سیستم کاربر استفاده می‌شود. آدرس ایمیل و رمز عبور کاربر را جمع‌آوری می‌کند. داده‌ها را برای اطمینان از اینکه ایمیل وجود دارد و رمز عبور صحیح است، اعتبارسنجی می‌کند.

UpdateProfileForm: این فرم برای به‌روزرسانی پروفایل کاربر استفاده می‌شود. نام کاربری و آدرس ایمیل کاربر را جمع‌آوری می‌کند. همچنین داده‌ها را برای اطمینان از اینکه نام کاربری منحصربه‌فرد است، ایمیل معتبر است و کاربر در حال تلاش برای به‌روزرسانی پروفایل خود است، اعتبارسنجی می‌کند.

PostForm: این فرم برای ایجاد پست‌های جدید استفاده می‌شود. عنوان و محتوای پست را جمع‌آوری می‌کند. داده‌ها را برای اطمینان از اینکه عنوان خالی نیست و محتوا خالی نیست، اعتبارسنجی می‌کند.

اعتبارسنجی فرم

هر کلاس فرم از اعتبارسنج‌های WTForms برای اعتبارسنجی ورودی کاربر استفاده می‌کند. این اعتبارسنج‌ها اطمینان حاصل می‌کنند که داده‌ها در قالب مورد نیاز قرار دارند و معیارهای خاصی را برآورده می‌کنند. به عنوان مثال، RegistrationForm از DataRequired() برای اطمینان از پر شدن همه فیلدها، Length() برای محدود کردن طول نام کاربری بین 4 تا 25 کاراکتر و EqualTo() برای مقایسه فیلدهای رمز عبور و تأیید رمز عبور استفاده می‌کند.

ارث‌گیری فرم

RegistrationForm و UpdateProfileForm از کلاس FlaskForm که قابلیت‌های اساسی فرم را فراهم می‌کند، ارث می‌برند. LoginForm از FlaskForm ارث می‌برد و یک فیلد remember را برای اجازه دادن به کاربران برای ماندن در حالت ورود به سیستم اضافه می‌کند. PostForm از FlaskForm ارث می‌برد و هیچ قابلیت اضافی اضافه نمی‌کند.

ادغام با Flask-Login

LoginForm و UpdateProfileForm از Flask-Login برای ادغام با سیستم احراز هویت کاربر برنامه استفاده می‌کنند. آنها از متغیر current_user برای دسترسی به کاربر فعلی استفاده می‌کنند و از آن برای انجام اعتبارسنجی‌های اضافی استفاده می‌کنند.

RegistrationForm

فیلد username باید بین 4 تا 25 کاراکتر طول داشته باشد.

فیلد email باید یک آدرس ایمیل معتبر باشد.

فیلدهای password و `confirm

فولدر templates:

درون فولدر templates چندین فایل html وجود دارد.

که به تفکیک به آنها میپردازیم.

فایل base.html :

این فایل فایل اصلی ما توی کدهای اچ تی ام ال محسوب میشود و باقی فایل ها از این فایل ارث بری میکنند(به وسیله کدهای جینجا2)

نکته: درون فایل base کدهای فایل navbar.html include میشوند.

یعنی هرچه در نوبار وجود دارد در فایل بیس ایجاد شده و فایل بیس هم به سایر فایل ها این ویژگی را به ارث میگذارد.

توضیحات بیشتر:

عنوان صفحه: این عنوان می‌تواند توسط هر صفحه دیگری در برنامه با استفاده از دستور {{ page_title }} تغییر داده شود.

کتابخانه‌های Bootstrap و Font Awesome: این کتابخانه‌ها می‌توانند برای سفارشی‌سازی ظاهر و احساس برنامه استفاده شوند.

فایل styles.css: این فایل می‌تواند برای سفارشی‌سازی بیشتر ظاهر و احساس برنامه استفاده شود.

عنصر با ویژگی dir="rtl": این ویژگی نشان می‌دهد که محتوا باید به صورت راست به چپ (RTL) رندر شود.

فایل قالب navbar.html: این فایل می‌تواند برای سفارشی‌سازی نوار پیمایش برنامه استفاده شود.

فایل قالب messages.html: این فایل می‌تواند برای سفارشی‌سازی نحوه نمایش پیام‌ها و اعلان‌ها در برنامه استفاده شود.

بلوک {% block content %}{% endblock content %}: این بلوک یک جای خالی برای محتوای پویا هر صفحه است.

فایل home.html :

گسترش قالب پایه: عبارت {٪ extends 'base.html' ٪} نشان می‌دهد که home.html ساختار و طرح را از قالب base.html به ارث می‌برد. این امر یکنواختی را در همه صفحات برنامه تضمین می‌کند.

صفحه بزرگ: عنصر یک صفحه بزرگ با طرح متمرکز ایجاد می‌کند. در داخل، برچسب <h1 class="display-4">شعر...</h1> یک عنوان بزرگ را نمایش می‌دهد و برچسب <h3>{{ current_user.username }}</h3> نام کاربری کاربر وارد شده را نشان می‌دهد.

کارت‌های شعر: عنصر یک کارت‌هولدر با جهت راست به چپ (RTL) ایجاد می‌کند. در داخل، حلقه {% for post in posts %} از طریق یک لیست از اشیاء post که شعرها را نشان می‌دهد، تکرار می‌شود. برای هر پست، یک برچسب لنگر با کلاس card-body و محتوای عنوان پست ایجاد می‌شود. این پیوند کاربر را به صفحه‌ای که جزئیات شعر را نمایش می‌دهد، هدایت می‌کند.

JavaScript: برچسب <script src="/static/scripts.js"></script> فایل scripts.js را بارگذاری می‌کند، که احتمالاً حاوی کد JavaScript برای مدیریت رویدادها، انیمیشن‌ها یا سایر قابلیت‌های پویا خاص صفحه اصلی است.

کدهای جاوااسکریپت درون فایل scripts.js ذخیره شده‌اند.

کدهای سی اس اس درون فایل styles.css ذخیره شده‌اند.

فایل detail.html :

گسترش قالب پایه: عبارت {% extends 'base.html' %} نشان می‌دهد که detail.html ساختار و طرح را از قالب base.html به ارث می‌برد، که یکنواختی را در همه صفحات برنامه تضمین می‌کند.

عملیات پست مشروط: بلوک {% if post.author == current_user %} بررسی می‌کند که آیا کاربر فعلی نویسنده شعر است. اگر چنین است، دو دکمه برای حذف و ویرایش شعر با استفاده از برچسب‌های <a> با کلاس‌های btn btn-danger و btn btn-secondary به ترتیب نمایش داده می‌شوند.

عنوان پست: برچسب h3 عنوان شعر را نمایش می‌دهد، و برچسب p با کلاس post-meta نام کاربری نویسنده و تاریخ ایجاد شعر را فرمت‌شده با استفاده از روش strftime() نمایش می‌دهد.

محتوای پست: عنصر با کلاس post-content محتوای شعر را در خود نگه می‌دارد. عبارت {{ post.content.replace('\n', '
') | safe }} هر نویسه خط جدید (\n) را به برچسب‌های خط جدید HTML (
) تبدیل می‌کند، که قالب‌بندی مناسب را تضمین می‌کند.

دکمه‌های اشتراک‌گذاری: پنج دکمه اشتراک‌گذاری رسانه‌های اجتماعی با استفاده از برچسب‌های <a> با کلاس btn btn-info ایجاد می‌شوند که هر کدام حاوی یک آیکون از Font Awesome هستند. ویژگی target="_blank" پیوند اشتراک‌گذاری را در یک زبانه جدید مرورگر باز می‌کند. URLهای اشتراک‌گذاری با استفاده از url_for() و _external=True ایجاد می‌شوند، که اطمینان می‌دهد پیوندها به پلتفرم‌های مناسب اشاره دارند.

پلتفرم ها:

تلگرام، توییتر، لینکدین، واتس اپ و فیسبوک.

فایل create_post.html :

فرم HTML: فرم HTML با استفاده از دستور <form> تعریف می‌شود. ویژگی action فرم را به مسیر `` تنظیم می‌کند، که به معنی ارسال داده‌های فرم به خود برنامه است. ویژگی method فرم را به post تنظیم می‌کند، که نشان می‌دهد داده‌های فرم به روش HTTP POST ارسال می‌شوند.

فیلد csrf_token: فیلد csrf_token توسط Flask برای محافظت از برنامه در برابر حملات CSRF استفاده می‌شود. این فیلد باید با مقداری که توسط Flask ایجاد می‌شود پر شود.

فیلدهای ورودی متنی: فیلدهای ورودی متنی با استفاده از دستور <input> تعریف می‌شوند. ویژگی type فیلد را به text تنظیم می‌کند، که نشان می‌دهد این فیلد یک فیلد ورودی متنی است. ویژگی class فیلد را به form-control تنظیم می‌کند، که یک کلاس CSS است که ظاهر فیلد را تنظیم می‌کند. ویژگی placeholder فیلد را با یک متن پیش‌فرض پر می‌کند.

دکمه ارسال: دکمه ارسال با استفاده از دستور <input> تعریف می‌شود. ویژگی type دکمه را به submit تنظیم می‌کند، که نشان می‌دهد این دکمه یک دکمه ارسال است. ویژگی value دکمه را با متن ایجاد تنظیم می‌کند.

برای استفاده از این قالب، باید آن را به مسیری که می‌خواهید فرم در آن نمایش داده شود، اضافه کنید. سپس، می‌توانید از یک شیء Form برای ایجاد یک فرم با فیلدهای مورد نظر خود استفاده کنید.

فایل login.html:

فرم HTML: فرم HTML با استفاده از دستور <form> تعریف می‌شود. ویژگی action فرم را به مسیر `` تنظیم می‌کند، که به معنی ارسال داده‌های فرم به خود برنامه است. ویژگی method فرم را به post تنظیم می‌کند، که نشان می‌دهد داده‌های فرم به روش HTTP POST ارسال می‌شوند.

فیلد csrf_token: فیلد csrf_token توسط Flask برای محافظت از برنامه در برابر حملات CSRF استفاده می‌شود. این فیلد باید با مقداری که توسط Flask ایجاد می‌شود پر شود.

فیلدهای ورودی متنی: فیلدهای ورودی متنی با استفاده از دستور <input> تعریف می‌شوند. ویژگی type فیلد را به text تنظیم می‌کند، که نشان می‌دهد این فیلد یک فیلد ورودی متنی است. ویژگی class فیلد را به form-control تنظیم می‌کند، که یک کلاس CSS است که ظاهر فیلد را تنظیم می‌کند. ویژگی placeholder فیلد را با یک متن پیش‌فرض پر می‌کند.

کادر چک: کادر چک با استفاده از دستور <input> تعریف می‌شود. ویژگی type کادر چک را به checkbox تنظیم می‌کند. ویژگی class کادر چک را به form-check-input تنظیم می‌کند، که یک کلاس CSS است که ظاهر کادر چک را تنظیم می‌کند. ویژگی checked کادر چک را به مقدار checked تنظیم می‌کند تا کادر چک به طور پیش‌فرض انتخاب شود.

دکمه ارسال:

دکمه ارسال با استفاده از دستور <input> تعریف می‌شود. ویژگی type دکمه را به submit تنظیم می‌کند، که نشان می‌دهد این دکمه یک دکمه ارسال است. ویژگی value دکمه را با متن «ورود» تنظیم می‌کند.

نحوه استفاده:

برای استفاده از این قالب، باید آن را به مسیری که می‌خواهید فرم در آن نمایش داده شود، اضافه کنید. سپس، می‌توانید از یک شیء Form برای ایجاد یک فرم با فیلدهای مورد نظر خود استفاده کنید.

برای مثال، می‌توانید کد زیر را برای ایجاد یک فرم با دو فیلد ورودی متنی استفاده کنید:

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField

class LoginForm(FlaskForm):
    email = StringField('ایمیل')
    password = PasswordField('رمز عبور')
```

سپس، می‌توانید این فرم را در قالب login.html استفاده کنید:

```
{٪ 'extends 'base.html ٪}

{٪ block content ٪}
<"form action="" method="post>
{{ form.csrf_token }}

<"div class="form-group>
{{ (form.email.label(class='form-control-label', text='ایمیل') }}
{{ form.email(class='form-control', placeholder='آدرس ایمیل را وارد کنید') }}
<div/>

<"div class="form-group>
{{ (form.password.label(class='form-control-label', text='رمز عبور') }}
{{ form.password(class='form-control', placeholder='رمز عبور خود را وارد کنید') }}
<div/>

<input type="submit" value="ورود" class="btn btn-info">
<form/>
{٪ endblock ٪}
```

با استفاده از این کد، یک فرم با دو فیلد ورودی متنی به نام‌های email و password نمایش داده می‌شود. کاربران می‌توانند اطلاعات خود را در این فیلدها وارد کنند و سپس بر روی دکمه «ورود» کلیک کنند

فایل profile.html :

یک قالب HTML است که صفحه پروفایل کاربر را در یک برنامه Flask تعریف می‌کند. این فایل از قالب base.html گسترش می‌یابد.

دکمه ایجاد پست جدید: این دکمه با استفاده از دستور <a> تعریف می‌شود. ویژگی href دکمه را به مسیر new_post تنظیم می‌کند، که به معنی هدایت کاربر به صفحه ایجاد پست جدید است. ویژگی class دکمه را به btn btn-primary btn-lg تنظیم می‌کند، که یک کلاس CSS است که ظاهر دکمه را تنظیم می‌کند.

اطلاعات کاربر فعلی: اطلاعات کاربر فعلی با استفاده از عبارت {{ current_user.username }} و {{ current_user.email }} نمایش داده می‌شوند. این عبارت‌ها از شیء current_user که توسط Flask در دسترس قرار داده می‌شود، استفاده می‌کنند.

فرم به‌روزرسانی اطلاعات کاربر: این فرم با استفاده از دستور <form> تعریف می‌شود. ویژگی action فرم را به مسیر `` تنظیم می‌کند، که به معنی ارسال داده‌های فرم به خود برنامه است. ویژگی method فرم را به post تنظیم می‌کند، که نشان می‌دهد داده‌های فرم به روش HTTP POST ارسال می‌شوند.

فایل register.html:

بخش نمایش خطاهای فرم: این بخش با استفاده از عبارت {٪ 'include 'inc/form_errors.html ٪} شامل خطاهای فرم می‌شود. این خطاها توسط کتابخانه WTForms ایجاد می‌شوند و در صورت بروز مشکل در فرم، نمایش داده می‌شوند.

فرم ثبت نام: این فرم با استفاده از دستور <form> تعریف می‌شود. ویژگی action فرم را به مسیر `` تنظیم می‌کند، که به معنی ارسال داده‌های فرم به خود برنامه است. ویژگی method فرم را به post تنظیم می‌کند، که نشان می‌دهد داده‌های فرم به روش HTTP POST ارسال می‌شوند.

فیلد username: این فیلد با استفاده از دستور <input type="text"> تعریف می‌شود. ویژگی name فیلد را به username تنظیم می‌کند، که نام این فیلد در مدل کاربر است. ویژگی class فیلد را به form-control تنظیم می‌کند، که یک کلاس CSS است که ظاهر فیلد را تنظیم می‌کند. ویژگی placeholder فیلد را با یک متن پیش‌فرض پر می‌کند.

فیلد email: این فیلد با استفاده از دستور <input type="text"> تعریف می‌شود. ویژگی‌های name, class, و placeholder این فیلد مشابه فیلد username است.

فیلد password: این فیلد با استفاده از دستور <input type="password"> تعریف می‌شود. ویژگی‌های name, class, و placeholder این فیلد مشابه فیلد username است.

فیلد confirm_password: این فیلد با استفاده از دستور <input type="password"> تعریف می‌شود. ویژگی‌های name, class, و placeholder این فیلد مشابه فیلد password است. این فیلد برای تأیید رمز عبور کاربر استفاده می‌شود.

با استفاده از این کد، یک فرم با چهار فیلد ورودی متنی به نام‌های username، email، password، و confirm_password نمایش داده می‌شود.

فایل update.html :

این فایل از قالب base.html گسترش می‌یابد.

فرم بروزرسانی: فرم بروزرسانی با استفاده از دستور <form> تعریف می‌شود. ویژگی action فرم را به مسیر `` تنظیم می‌کند، که به معنی ارسال داده‌های فرم به خود برنامه است. ویژگی method فرم را به post تنظیم می‌کند، که نشان می‌دهد داده‌های فرم به روش HTTP POST ارسال می‌شوند.

فیلدهای ورودی متنی: فیلدهای ورودی متنی با استفاده از دستور <input type="text"> تعریف می‌شوند. ویژگی class فیلدها را به form-control تنظیم می‌کند، که یک کلاس CSS است که ظاهر فیلدها را تنظیم می‌کند. ویژگی placeholder فیلدها را با یک متن پیش‌فرض پر می‌کند.

دکمه ارسال: دکمه ارسال با استفاده از دستور <input type="submit"> تعریف می‌شود. ویژگی value دکمه را به «بروزرسانی» تنظیم می‌کند.

با استفاده از این کد، یک فرم با دو فیلد ورودی متنی به نام‌های title و content نمایش داده می‌شود. کاربران می‌توانند اطلاعات خود را در این فیلدها وارد کنند و سپس بر روی دکمه «بروزرسانی» کلیک کنند تا پست به‌روزرسانی شود.

درون فولدر templates یک فولدر داریم به نام inc که مخفف include است.

به تفکیک فایل‌های اچ تی ام درون ان را بررسی میکنیم.

فایل navbar.html :

نوار پیمایش تیره: برچسب nav یک نوار پیمایش تیره با موقعیت ثابت در بالا را تعریف می‌کند. کلاس navbar-expand-lg نوار پیمایش را واکنش‌گرا می‌کند و آن را در صفحه‌نمایش‌های بزرگتر گسترش می‌دهد. کلاس navbar-dark رنگ پس‌زمینه را به خاکستری تیره و کلاس bg-dark رنگ متن را به سفید تنظیم می‌کند.

تصویر لوگو: برچسب a با کلاس navbar-brand به صفحه اصلی پیوند می‌دهد و شامل یک تصویر لوگو می‌شود. ویژگی src برچسب img مسیر به تصویر لوگو را مشخص می‌کند.

دکمه تغییر اندازه: برچسب button با کلاس navbar-toggler یک دکمه تغییر اندازه ایجاد می‌کند که به کاربران اجازه می‌دهد نوار پیمایش را جمع کنند. ویژگی data-toggle دکمه را به نوع collapse تنظیم می‌کند، و ویژگی data-target ID عنصری را که باید جمع شود مشخص می‌کند (در این مورد، #navbarNav). ویژگی aria-controls ID عنصری را که محتوای جمع‌شده را در خود جای می‌دهد مشخص می‌کند (در این مورد، navbarNav)، و ویژگی aria-expanded نشان می‌دهد که آیا محتوا در حال حاضر گسترش یافته است (در این مورد، نادرست).

پیوندهای پیمایش: برچسب ul با کلاس navbar-nav یک لیست از پیوندهای پیمایش را تعریف می‌کند. برچسب‌های li با کلاس nav-item موارد پیمایش فردی را ایجاد می‌کنند، و برچسب‌های a با کلاس nav-link پیوندهای ابرمتن به صفحات مربوطه ایجاد می‌کنند. برچسب‌های i با آیکون‌های Font Awesome برای افزودن نشانه‌های بصری به پیوندها استفاده می‌شوند.

تصدیق کاربر: بلوک‌های {٪ if current_user.is_authenticated ٪} و {٪ endif ٪} پیوندهای پیمایش را بر اساس وضعیت تأیید اعتبار کاربر به صورت مشروط نمایش می‌دهند. هنگامی که کاربر وارد سیستم شده است، پیوندهای پروفایل، خروج، و ورود با گوگل نمایش داده می‌شوند. هنگامی که کاربر وارد سیستم نشده است، پیوندهای ثبت نام، ورود، و ورود با گوگل نمایش داده می‌شوند.

ویژگی href برچسب a پیوندهای پیمایش را به مسیرهای مربوطه در برنامه Flask پیوند می‌دهد.

ویژگی class برچسب‌های li کلاس‌های CSS را به عناصر پیمایش اضافه می‌کند. این کلاس‌ها می‌توانند برای تنظیم ظاهر عناصر پیمایش استفاده شوند.

ویژگی alt برچسب img توضیحات جایگزین برای تصویر لوگو را فراهم می‌کند. این توضیحات در صورتی که تصویر لوگو به دلایلی بارگیری نشود، نمایش داده می‌شود.

فایل form_errors.html:

عنصر ظرف: برچسب div با کلاس container به عنوان ظرفی برای پیام‌های خطا عمل می‌کند. این ظرف به گروه‌بندی پیام‌های خطا کمک می‌کند و جداسازی بصری واضحی از بقیه فرم ایجاد می‌کند.

عناصر آلرت: برچسب‌های div با کلاس alert alert-danger alert-dismissible fade show برای نمایش پیام‌های خطای فردی استفاده می‌شوند. این آلرت‌ها دارای ویژگی‌های زیر هستند:

class: این کلاس سبک آلرت را به خطرناک تعریف می‌کند که نشان‌دهنده یک خطای جدی است.

role: این ویژگی مشخص می‌کند که آلرت یک جعبه گفتگو است.

dismissable: این ویژگی باعث می‌شود که آلرت با کلیک روی دکمه بستن قابل بستن باشد.

fade: این ویژگی باعث می‌شود که آلرت هنگام نمایش محو شود و ظاهر شود.

show: این ویژگی آلرت را در ابتدا نشان می‌دهد.

پیام‌های خطا: برچسب‌های strong داخل آلرت‌ها پیام‌های خطای واقعی را نمایش می‌دهند. این پیام‌ها توسط سیستم اعتبارسنجی فرم Flask ایجاد می‌شوند و خطاهای خاص را که با فیلدهای فرم مربوطه مواجه شده‌اند را نشان می‌دهند.

دکمه بستن: برچسب button با کلاس btn-close برای هر آلرت یک دکمه بستن ایجاد می‌کند. این دکمه را می‌توان برای بستن آلرت و پنهان کردن پیام خطا کلیک کرد.

برای استفاده از فایل form_errors.html، باید آن را در قالبی که فرم را نمایش می‌دهد، وارد کنید. سپس می‌توانید پیام‌های خطا را برای هر فیلد فرمی که دارای خطا است، به صورت مشروط با استفاده از برچسب قالب {٪ if form.field.errors ٪} رندر کنید.

فایل messages.html:

یک فایل قالب است که برای نمایش پیام‌های فلاش در یک برنامه Flask استفاده می‌شود. پیام‌های فلاش پیام‌های موقتی هستند که برای مدت کوتاهی برای کاربر نمایش داده می‌شوند.

فایل messages.html از برچسب قالب {٪ with messages = get_flashed_messages(with_categories=true) ٪} برای بازیابی تمام پیام‌های فلاش از برنامه Flask استفاده می‌کند. آرگومان with_categories=true اطمینان حاصل می‌کند که پیام‌ها با دسته‌های مرتبط خود بازگردانده می‌شوند.

اگر هرگونه پیام فلاش وجود داشته باشد، قالب از حلقه {٪ for cat, msg in messages ٪} برای تکرار آنها استفاده می‌کند و دسته و محتوای پیام را استخراج می‌کند. برای هر پیام، یک عنصر آلرت < div class="alert alert-{{ cat }} alert-dismissible fade show" role="alert"> با کلاس دسته‌بندی مربوطه و محتوای پیام ایجاد می‌کند.

یک دکمه بستن اختیاری <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button> به هر عنصر آلرت اضافه می‌شود که به کاربران امکان می‌دهد پیام را رد کنند.

در نهایت، قالب عناصر آلرت را در یک عنصر ظرف و یک عنصر قرار می‌دهد تا ساختار بصری واضحی برای نمایش پیام‌های فلاش فراهم کند.

درون فولدر blog یک فولدر دیگری داریم به نام static که درون آن فولدری وجود دارد به نام images که تصاویر را ذخیره میکند و در کنار این فولدر دوتا فایل styles.css و scripts.js را داریم که به ترتیب کدهای سی اس اس و جاواسکریپت را درون خود ذخیره میکند.

فایل scripts.js:

فایل scripts.js حاوی کد جاوا اسکریپت است که رفتار موس را برای کارت پستال‌ها در یک صفحه وب مدیریت می‌کند. از متد document.addEventListener() برای افزودن شنونده‌های رویداد به هر عنصر کارت پستال استفاده می‌کند و بر اساس تعاملات ماوس، سبک‌های CSS را اضافه یا حذف می‌کند.

شنونده رویداد DOMContentLoaded:

عبارت document.addEventListener("DOMContentLoaded", function(){}) اطمینان حاصل می‌کند که کد جاوا اسکریپت پس از بارگیری و تجزیه کامل سند HTML اجرا می‌شود. این اطمینان حاصل می‌کند که عناصر کارت پستال برای دستکاری در دسترس هستند.

انتخاب کارت پستال‌ها:

عبارت var postCards = document.querySelectorAll('.post-card'); همه عناصر با کلاس post-card را انتخاب می‌کند و آنها را در آرایه postCards ذخیره می‌کند. این امکان را می‌دهد که کد هر عنصر کارت پستال را به طور جداگانه تکرار و دستکاری کند.

شنونده رویداد ماوس وارد شدن:

حلقه postCards.forEach(function(card) {...}) بر روی هر عنصر کارت پستال در آرایه postCards تکرار می‌شود. برای هر عنصر، از متد card.addEventListener('mouseenter', function() {...}) برای افزودن یک شنونده رویداد mouseenter استفاده می‌کند.

انیمیشن و سایه موس:

داخل هدر رویداد mouseenter، کد از عبارات this.style.transform = 'scale(1.05)'; و this.style.boxShadow = '0 8px 16px rgba(0, 0, 0, 0.2)'; برای اعمال سبک‌های CSS به عنصر کارت پستال استفاده می‌کند. این باعث می‌شود که کارت پستال کمی مقیاس شود و یک سایه ریز ایجاد کند، که یک اثر موس ظریف ایجاد می‌کند.

شنونده رویداد ماوس خارج شدن:

به طور مشابه، کد از متد card.addEventListener('mouseleave', function() {...}) برای افزودن یک شنونده رویداد mouseleave به هر کارت پستال استفاده می‌کند. وقتی ماوس از کارت پستال خارج می‌شود، کد سبک‌های CSS را با استفاده از this.style.transform = 'scale(1)'; و this.style.boxShadow = '0 4px 8px rgba(0, 0, 0, 0.1)'; به حالت اولیه خود بازمی‌گرداند.

اثر کلی:

ترکیب شنونده رویدادهای mouseenter و mouseleave یک اثر موس روان و جذاب بصری را برای کارت پستال‌ها ایجاد می‌کند. وقتی ماوس روی کارت پستال قرار می‌گیرد، کمی بزرگ می‌شود و یک سایه اضافه می‌کند، که نشان می‌دهد که عنصر در حال تعامل است. وقتی ماوس خارج می‌شود، کارت پستال به حالت اولیه خود بازمی‌گردد. این رفتار تجربه کاربر را بهبود می‌بخشد و لایه‌ای از تعامل را به صفحه وب اضافه می‌کند.

فایل styles.css:

یک فایل CSS است که برای استایل دادن به صفحات detail.html و home.html استفاده می‌شود.

در صفحه detail.html، فایل styles.css استایل‌های زیر را اعمال می‌کند:

رنگ پس زمینه صفحه را به یک تصویر ثابت با نام aser.jpg تغییر می‌دهد.

رنگ متن صفحه را به #333 تغییر می‌دهد.

برای عنصر .post-details حاشیه، پدینگ، حاشیه، شعاع گوشه، سایه و رنگ پس زمینه را تنظیم می‌کند.

برای عنصر .post-actions حاشیه پایین را تنظیم می‌کند.

برای عناصر .post-actions a حاشیه راست را تنظیم می‌کند.

برای عنصر .post-header h3 رنگ متن، اندازه فونت، حاشیه پایین و پدینگ را تنظیم می‌کند.

برای عنصر .post-meta رنگ متن و حاشیه پایین را تنظیم می‌کند.

برای عنصر .post-content p ارتفاع خط و اندازه فونت را تنظیم می‌کند.

برای عنصر .share-buttons حاشیه بالا را تنظیم می‌کند.

برای عناصر .share-buttons a حاشیه راست، رنگ متن، عدم استفاده از خط زیر و اندازه فونت را تنظیم می‌کند.

برای عناصر .share-buttons a:hover شفافیت را تنظیم می‌کند.

استایل‌های صفحه home.html

در صفحه home.html، فایل styles.css استایل‌های زیر را اعمال می‌کند:

برای عنصر .home فونت را تنظیم می‌کند.

برای عنصر .home h3 رنگ متن، متن‌چینی، پدینگ، رنگ پس زمینه، حاشیه بالا و پایین، فونت و شفافیت را تنظیم می‌کند.

برای عنصر .home .card حاشیه، پدینگ، رنگ پس زمینه، حاشیه، شعاع گوشه، سایه، انتقالات و شفافیت را تنظیم می‌کند.

برای عنصر .home .card:hover مقیاس، سایه و شفافیت را تنظیم می‌کند.

برای عنصر .home .card-body رنگ متن، نمایش، پدینگ، حاشیه، حاشیه، شعاع گوشه، انتقالات و رنگ پس زمینه را تنظیم می‌کند.

برای عنصر .home .card-body:hover رنگ پس زمینه را تنظیم می‌کند.

چند نمونه از نحوه استفاده از استایل‌های اعمال شده:

برای تغییر رنگ پس زمینه صفحه detail.html می‌توانید مقدار background-color را به رنگ دلخواه خود تغییر دهید.

برای افزایش اندازه فونت عنصر .post-content p می‌توانید مقدار font-size را افزایش دهید.

برای افزودن یک حاشیه به عنصر .home .card می‌توانید مقدار border را تنظیم کنید.

برای تغییر رنگ سایه عنصر .home .card می‌توانید مقدار box-shadow را تنظیم کنید.

درون فولدر blog ما پایگاه داده ای ایجاد کردیم به نام blog.db.

فایل blog.db یک پایگاه داده SQLite است که برای ذخیره اطلاعات مربوط به پست‌های وبلاگ استفاده می‌شود. این پایگاه داده حاوی دو جدول است:

posts که اطلاعات مربوط به هر پست وبلاگ را ذخیره می‌کند.

tags که اطلاعات مربوط به برچسب‌های هر پست وبلاگ را ذخیره می‌کند.

جدول posts شامل ستون‌های زیر است:

id که یک شناسه منحصربه‌فرد برای هر پست وبلاگ است.

title که عنوان پست وبلاگ را ذخیره می‌کند.

content که محتوای پست وبلاگ را ذخیره می‌کند.

created_at که تاریخ و زمان ایجاد پست وبلاگ را ذخیره می‌کند.

updated_at که تاریخ و زمان به‌روزرسانی پست وبلاگ را ذخیره می‌کند.

جدول tags شامل ستون‌های زیر است:

id که یک شناسه منحصربه‌فرد برای هر برچسب است.

name که نام برچسب را ذخیره می‌کند.

با استفاده از پایگاه داده، می‌توانید به راحتی اطلاعات مربوط به پست‌های وبلاگ خود را ذخیره، بازیابی، به‌روزرسانی و حذف کنید.

فایل requeriments.txt:

فایل requirements.txt یک فایل متنی است که فهرستی از تمام بسته‌های Python است که یک پروژه به آن‌ها وابسته است. این فایل توسط ابزار مدیریت بسته pip برای نصب بسته‌های ضروری هنگام نصب پروژه استفاده می‌شود.

چارچوب وب و بسته‌های مرتبط

Flask: یک چارچوب وب Python

Flask-Bcrypt: یک افزونه Flask برای مدیریت رمزهای عبور هش شده bcrypt

Flask-JWT-Extended: یک افزونه Flask برای پیاده‌سازی توکن‌های JSON Web (JWT)

Flask-Login: یک افزونه Flask برای مدیریت ورود کاربران

Flask-Migrate: یک افزونه Flask برای مدیریت مهاجرت‌های پایگاه داده

Flask-OAuthlib: یک افزونه Flask برای ادغام با OAuth2

Flask-Reuploaded: یک افزونه Flask برای مدیریت بارگذاری فایل با پشتیبانی از بازیابی

Flask-SQLAlchemy: یک افزونه Flask برای کار با SQLAlchemy

Flask-Uploads: یک افزونه Flask برای مدیریت بارگذاری فایل

Flask-WTF: یک افزونه Flask برای ایجاد فرم‌ها

بسته‌های امنیت و احراز هویت

absl-py: یک کتابخانه کمکی Python برای NumPy و TensorFlow

alembic: یک ابزار مهاجرت پایگاه داده

attrs: یک کتابخانه Python برای ایجاد کلاس‌های داده

Authlib: یک کتابخانه Python برای پیاده‌سازی OAuth2 و OpenID Connect

bcrypt: یک کتابخانه C برای هش کردن رمزهای عبور به‌طور ایمن

cachelib: یک کتابخانه Python برای ذخیره‌سازی داده‌ها

cachetools: یک کتابخانه Python برای ذخیره‌سازی داده‌ها با پشته‌های قابل پیکربندی

certifi: یک کتابخانه Python برای تأیید گواهینامه‌های HTTPS

cffi: یک کتابخانه Python برای جاسازی افزونه‌های C

charset-normalizer: یک کتابخانه Python برای نرمال‌سازی متن Unicode

click: یک کتابخانه Python برای ایجاد رابط‌های خط فرمان

colorama: یک کتابخانه Python برای رنگ‌آمیزی متن در برنامه‌های خط فرمان

cryptography: یک کتابخانه Python برای الگوریتم‌ها و پروتکل‌های رمزنگاری

cycler: یک کتابخانه Python برای تولید طرح‌های رنگی

decorator: یک کتابخانه Python برای تزئین توابع

dnspython: یک کتابخانه Python برای کار با DNS

email-validator: یک کتابخانه Python برای اعتبارسنجی آدرس‌های ایمیل

بسته‌های دیگر

fonttools: یک کتابخانه Python برای کار با فونت‌ها

google-auth: یک کتابخانه Python برای احراز هویت با APIهای Google

google-auth-oauthlib: یک کتابخانه Python برای استفاده از Google OAuth2 برای احراز هویت

greenlet: یک کتابخانه Python برای کار با کارگزاری‌ها

httplib2: یک کتابخانه Python برای ارسال درخواست‌های HTTP

idna: یک کتابخانه Python برای نام‌های دامنه بین‌المللی

importlib-metadata: یک کتابخانه Python برای دسترسی به متاداده بسته‌ها

importlib-resources: یک کتابخانه Python برای کار با منابع واردات

is-disposable-email: یک کتابخانه Python برای بررسی اینکه آیا یک آدرس ایمیل قابل‌تخصیص است

itsdangerous: یک کتابخانه Python برای ایجاد توکن‌های URL ایمن

Jinja2: یک کتابخانه Python برای قالب‌بندی

kiwisolver: یک کتابخانه Python برای حل سیستم‌های معادلات خطی

Mako: یک موتور قالب‌بندی Python

MarkupSafe: یک کتابخانه Python برای فرار HTML

matplotlib: یک کتابخانه Python برای ایجاد نمودارهای 2 بعدی

mediapipe: یک کتابخانه Python برای کار با بینایی رایانه

MouseInfo: یک کتابخانه Python برای دریافت اطلاعات در مورد ماوس

numpy: یک کتابخانه Python برای محاسبات علمی

oauth2client: یک کتابخانه Python برای احراز هویت OAuth2

oauthlib: یک کتابخانه Python برای OAuth2

Pillow: یک کتابخانه Python برای کار با تصاویر

pip: یک ابزار مدیریت بسته Python

protobuf: یک کتابخانه Python برای کار با پروتکل Buffers

پایان‌بندی:

از شما که وقت خود را برای خواندن این مستندات صرف کردید، متشکریم. امیدواریم که این مستندات برای شما مفید بوده باشد و از چند شعری که از خودم درون وبلاگ منتشر کردم لذت ببرید.