from flask import Flask, render\_template,flash,redirect,url\_for,session,logging,request

from flask\_mysqldb import MySQL

from wtforms import Form,StringField,TextAreaField,PasswordField,validators

from passlib.hash import sha256\_crypt

from functools import wraps # decorator larda kullanılıyor.

# Kullanıcının Giriş yapıp yapmadığını kontrol eden decorator. Flask dan alındı

def login\_required(f): # decorator bir kez yazılır ve gerekli olan her yerde kullanılabilir

@wraps(f)

def decorated\_function(\*args, \*\*kwargs):

if "logged\_in" in session: # eğer session başlatıldıysa (giriş yapılmışsa)

return f(\*args, \*\*kwargs) # dashboard çalıştırılır.

else:

flash("Bu Sayfayı Görüntülemek/Silmek için Lütfen Giriş Yapınız.","danger" )

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

return decorated\_function

# Kullanıcı Kayıt Formu

class RegisterForm(Form):

name = StringField("İsim Soyisim",validators=[validators.Length(min = 4, max =25, message = "lütfen geçerli İsim Soyisim giriniz")])

username = StringField("Kullanıcı Adı",validators=[validators.Length(min = 5, max =35, message = "lütfen min. 5 ve mak. 35 karakterden oluşan bir Kullanıcı Adı giriniz")])

email = StringField("Email Adresi",validators=[validators.Email(message = "Lütfen Geçerli bir Email Adresi Giriniz.")])

password = PasswordField("Parola:",validators=[

validators.DataRequired(message ="Lütfen bir Parola Belirleyin"),

validators.EqualTo(fieldname = "confirm",message = "Parolanızı Yanlış Girdiniz")

])

confirm = PasswordField("Parolanızı Doğrulayın")

class LoginForm(Form):

username = StringField("Kullanıcı Adı")

password = PasswordField("Parola")

app = Flask(\_\_name\_\_)

app.secret\_key = "ybblog" # flash mesajlarının yayınlanması için gerekli

app.config["MYSQL\_HOST"]= "localhost"

app.config["MYSQL\_USER"]= "root"

app.config["MYSQL\_PASSWORD"]= ""

app.config["MYSQL\_DB"]= "ybblog"

app.config["MYSQL\_CURSORCLASS"]= "DictCursor" # veriler sözlük yapısı halinde gelir

mysql = MySQL(app)

@app.route("/")

def index():

return render\_template("index.html")

@app.route("/about")

def about():

return render\_template("about.html")

@app.route("/dashboard") # login yapılmamış (session False) ise dashbord fonksiyonu çalıştırılmaz.

@login\_required # dashboard fonksiyonu çalıştırılmadan hemen önce decorator çalıştırılır

def dashboard():

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles where author = %s"

result = cursor.execute(sorgu,(session["username"],))

if result > 0:

articles = cursor.fetchall()

return render\_template("dashboard.html",articles = articles)

else:

return render\_template("dashboard.html")

return render\_template("dashboard.html")

# Kayıt Olma

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

def register():

form = RegisterForm(request.form)

if request.method == "POST" and form.validate():

name = form.name.data

username = form.username.data

email = form.email.data

password = sha256\_crypt.encrypt(form.password.data)

cursor = mysql.connection.cursor()

sorgu = "Insert into users(name,email,username,password) VALUES(%s,%s,%s,%s)"

cursor.execute(sorgu,(name,email,username,password))

mysql.connection.commit()

cursor.close()

flash("Tebrikler! Kaydınız Gerçekleşti.","success") # bir sonraki request de çıkar

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

else:

return render\_template("register.html",form = form)

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

def login():

form = LoginForm(request.form)

if request.method == "POST": # giriş yap butonuna tıklandıysa

username = form.username.data # girilen parola

password\_entered = form.password.data # girilen password

cursor = mysql.connection.cursor()

sorgu = "Select \* From users where username = %s"

result = cursor.execute(sorgu,(username,)) # girilen kullanıcı adı veri tabanında var mı?

if result > 0: # varsa # kullanıcı yoksa result sıfır gelir

data = cursor.fetchone() # sadece ilgili kullanıcının tüm bilgileri alınır

real\_password = data["password"]

if sha256\_crypt.verify(password\_entered,real\_password): # verify fonksiyonu ile girilen ve gerçek şifre karşılatırılır.

flash("Başarıyla Giriş Yaptınız...","success")

session["logged\_in"] = True # oturum kontrolü sozluk yapısı şeklinde bir session değişkeni ile yapılır

session["username"] = username

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

else:

flash("Parolanızı Yanlış Girdiniz...","danger")

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

else: # yoksa

flash ("Böyle bir Kullanıcı Bulunamadı !","danger")

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

return render\_template("login.html",form = form)

# detay sayfası (makale arama)

@app.route("/article/") # makale id kaçsa o gelecek (dinamik url)

def article(id):

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles where id = %s"

result = cursor.execute(sorgu,(id,)) # article olabilir veya olmayabilir (sıfır)

if result > 0:

article = cursor.fetchone()

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

else:

return render\_template("article.html")

# logout işlemi

@app.route("/logout")

def logout():

session.clear() # session son buldu

return redirect(url\_for("index")) # ana sayfaya dönüş

# Dinamik url

@app.route("/article/")

def detail(id):

return "Artical Id:" + id

# Makale Ekleme

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

@login\_required

def addarticle():

form = ArticleForm(request.form)

if request.method == "POST" and form.validate(): # Makale Ekle butonunu tıkladıysak ve formun validasyonu True ise

title = form.title.data # title verisini alır

content = form.content.data # content verisini alır

cursor = mysql.connection.cursor()

sorgu = "Insert into articles(title,author,content) VALUES (%s,%s,%s)"

cursor.execute(sorgu,(title,session["username"],content)) # username değerini session dan alır.

mysql.connection.commit()

cursor.close()

flash("Makale Başarı ile Eklendi","success")

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

return render\_template("addarticle.html",form = form)

# Makale Formu Oluşturma

class ArticleForm(Form):

title = StringField("Makale Başlığı",validators=[validators.Length(min=5,max=50, message= "Makale Başlığı Beş Karakterden Az Olamaz !")])

content = TextAreaField("Makale İçeriği", validators=[validators.Length(min=10, message = "Makale İçeriği 10 Karakterden Az Olamaz !")])

# Makaleler Sayfası

@app.route("/articles")

def articles():

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles"

result = cursor.execute(sorgu) # makale hiç yoksa result sıfır gelir

if result > 0:

articles = cursor.fetchall() # tüm makaleleri sozluk yapısı şeklinde getirir

return render\_template("articles.html",articles = articles) # tüm makaleleri ilgili sayfaya getirir

else:

return render\_template("articles.html")

# Makale Silme

@app.route("/delete/")

@login\_required

def delete(id):

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles where author = %s and id = %s"

result = cursor.execute(sorgu,(session["username"],id))

if result > 0: # eğer hem yazar ve id varsa

sorgu2 = "Delete from articles where id = %s"

cursor.execute(sorgu2,(id,))

mysql.connection.commit()

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

else:

flash("Böyle bir Makale Yok veya Silme Yetkiniz Yok !","danger")

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

# Makale Güncelleme

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

@login\_required

def guncelle(id):

if request.method == "GET":

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles where id = %s and author = %s"

result = cursor.execute(sorgu,(id,session["username"]))

if result == 0: # makalenin hiç olmaması veya bize ait olmaması durumu

flash ("Böyle bir makale yok veya bu işleme yetkiniz yok","danger")

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

else:

article = cursor.fetchone()

form = ArticleForm() # form mevcut bilgilerden oluşturulduğu için (request.form) kullanılmadı

form.title.data = article["title"] # başlık verisini dataya yükler. Form mevcut bilgiler kullanılarak oluşturulur

form.content.data = article["content"] # içerik verisini dataya yükler

return render\_template("update.html",form = form) # form, update.html file a gönderilir.

else: # post request kısmı (makaleyi güncelle butonuna tıklandığında geçerli)

form = ArticleForm(request.form) # yeniden form oluşturulur.

newTitle = form.title.data

newContent = form.content.data

sorgu2 = "Update articles Set title = %s, content = %s where id = %s"

cursor = mysql.connection.cursor()

cursor.execute(sorgu2,(newTitle,newContent,id))

mysql.connection.commit()

flash (" Makale Başarıyla Güncellendi", "success")

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

# Arama URL

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

def search():

if request.method == "GET": # get request olduğunda ana sayfaya gitmesi için

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

else:

keyword = request.form.get("keyword") # arama çubuğundan aranan kelimeyi almak için

cursor = mysql.connection.cursor()

sorgu = "Select \* from articles where title like '%" + keyword + "%' "

result = cursor.execute(sorgu)

if result == 0:

flash("Aranan Kelimeye Uygun Makale Bulunamadı...","warning")

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

else:

articles = cursor.fetchall()

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

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)