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

from flask\_sqlalchemy import SQLAlchemy

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

app.config['DEBUG'] = True # displays runtime errors in the browser, too

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'mysql+pymysql://flicklist:password@localhost:8889/flicklist'

app.config['SQLALCHEMY\_ECHO'] = True

db = SQLAlchemy(app)

class User(db.Model):

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

email = db.Column(db.String(120), unique=True)

password = db.Column(db.String(120))

def \_\_init\_\_(self, email, password):

self.email = email

self.password = password

def \_\_repr\_\_(self):

return '<User %r>' % self.email

class Movie(db.Model):

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

name = db.Column(db.String(120))

watched = db.Column(db.Boolean)

rating = db.Column(db.String(20))

def \_\_init\_\_(self, name):

self.name = name

self.watched = False

def \_\_repr\_\_(self):

return '<Movie %r>' % self.name

# a list of movie names that nobody should have to watch

terrible\_movies = [

"Gigli",

"Star Wars Episode 1: Attack of the Clones",

"Paul Blart: Mall Cop 2",

"Nine Lives",

"Starship Troopers"

]

def get\_current\_watchlist():

return Movie.query.filter\_by(watched=False).all()

def get\_watched\_movies():

return Movie.query.filter\_by(watched=True).all()

# TODO 3: Add "/login" GET and POST routes.

# TODO 4: Create login template with username and password.

# Notice that we've already created a 'login' link in the upper-right corner of the page that'll connect to it.

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

def register():

if request.method == 'POST':

email = request.form['email']

password = request.form['password']

verify = request.form['verify']

if not is\_email(email):

flash('zoiks! "' + email + '" does not seem like an email address')

return redirect('/register')

# TODO 1: validate that form value of 'verify' matches password

if password != verify:

flash("passwords don't match")

return redirect ('/')

# TODO 2: validate that there is no user with that email already

existing\_user = User.query.filter\_by(email=email).first()

if not existing\_user:

user = User(email=email, password=password)

db.session.add(user)

db.session.commit()

session['user'] = user.email

return redirect("/")

else:

flash('This user already exists')

return render\_template("register.html")

else:

return render\_template('register.html')

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

def login():

if request.method == 'POST':

email = request.form['email']

password = request.form['password']

user = User.query.filter\_by(email=email).first()

if user and user.password == password:

session['email'] = email

return redirect('/edit')

else:

flash('user password incorrect, or user does not exist')

return render\_template('login.html')

def is\_email(string):

# for our purposes, an email string has an '@' followed by a '.'

# there is an embedded language called 'regular expression' that would crunch this implementation down

# to a one-liner, but we'll keep it simple:

atsign\_index = string.find('@')

atsign\_present = atsign\_index >= 0

if not atsign\_present:

return False

else:

domain\_dot\_index = string.find('.', atsign\_index)

domain\_dot\_present = domain\_dot\_index >= 0

return domain\_dot\_present

@app.route("/logout", methods=['POST'])

def logout():

del session['user']

return redirect("/")

# Create a new route called rate\_movie which handles a POST request on /rating-confirmation

@app.route("/rating-confirmation", methods=['POST'])

def rate\_movie():

movie\_id = request.form['movie\_id']

rating = request.form['rating']

movie = Movie.query.get(movie\_id)

if movie not in get\_watched\_movies():

# the user tried to rate a movie that isn't in their list,

# so we redirect back to the front page and tell them what went wrong

error = "'{0}' is not in your Watched Movies list, so you can't rate it!".format(movie)

# redirect to homepage, and include error as a query parameter in the URL

return redirect("/?error=" + error)

# if we didn't redirect by now, then all is well

movie.rating = rating

db.session.add(movie)

db.session.commit()

return render\_template('rating-confirmation.html', movie=movie, rating=rating)

# Creates a new route called movie\_ratings which handles a GET on /ratings

@app.route("/ratings", methods=['GET'])

def movie\_ratings():

return render\_template('ratings.html', movies = get\_watched\_movies())

@app.route("/crossoff", methods=['POST'])

def crossoff\_movie():

crossed\_off\_movie\_id = request.form['crossed-off-movie']

crossed\_off\_movie = Movie.query.get(crossed\_off\_movie\_id)

if not crossed\_off\_movie:

return redirect("/?error=Attempt to watch a movie unknown to this database")

# if we didn't redirect by now, then all is well

crossed\_off\_movie.watched = True

db.session.add(crossed\_off\_movie)

db.session.commit()

return render\_template('crossoff.html', crossed\_off\_movie=crossed\_off\_movie)

@app.route("/add", methods=['POST'])

def add\_movie():

# look inside the request to figure out what the user typed

new\_movie\_name = request.form['new-movie']

# if the user typed nothing at all, redirect and tell them the error

if (not new\_movie\_name) or (new\_movie\_name.strip() == ""):

error = "Please specify the movie you want to add."

return redirect("/?error=" + error)

# if the user wants to add a terrible movie, redirect and tell them the error

if new\_movie\_name in terrible\_movies:

error = "Trust me, you don't want to add '{0}' to your Watchlist".format(new\_movie\_name)

return redirect("/?error=" + error)

movie = Movie(new\_movie\_name)

db.session.add(movie)

db.session.commit()

return render\_template('add-confirmation.html', movie=movie)

@app.route("/")

def index():

encoded\_error = request.args.get("error")

return render\_template('edit.html', watchlist=get\_current\_watchlist(), error=encoded\_error)

# TODO 5: modify this function to rely on a list of endpoints that users can visit without being redirected.

# It should contain 'register' and 'login'.

@app.before\_request

def require\_login():

allowed\_routes = ['register', 'login']

if not ('user' in session or request.endpoint in allowed\_routes):

return redirect("/register")

# In a real application, this should be kept secret (i.e. not on github)

# As a consequence of this secret being public, I think connection snoopers or

# rival movie sites' javascript could hijack our session and act as us,

# perhaps giving movies bad ratings - the HORROR.

app.secret\_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RU'

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

app.run()

Base.html

<html>

<head>

<title>FlickList</title>

<style type="text/css">

.error {

color: red;

}

</style>

</head>

<body>

<div style='float: right'>

{% if 'user' in session %}

{{ session['user'] }}

<form action="/logout" method="post">

<input type="submit" value="Log Out" />

</form>

{% else %}

<a href="login">login</a> | <a href="register">register</a>

{% endif %}

</div>

<h1>

<a href="/">FlickList</a>

</h1>

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<ul class=flashes>

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

{% block content %}

{% endblock %}

</body>

</html>

Edit.html

{% extends "base.html" %}

{% block content %}

{% if watchlist|length > 0 %}

<h2>My Watchlist</h2>

<ul>

{% for movie in watchlist: %}

<li>

<form action="/crossoff" method="post">

{{ movie.name }}

<input type="hidden" name="crossed-off-movie" value="{{ movie.id }}"/>

<input type="submit" value="I Watched it!"/>

</form>

</li>

{% endfor %}

</ul>

{% endif %}

<h2>Edit My Watchlist</h2>

<form action="/add" method="post">

<label>

I want to add

<input type="text" name="new-movie"/>

to my watchlist.

</label>

<input type="submit" value="Add It"/>

</form>

<h2><a href="/ratings">Ratings</a></h2>

{% if error %}

<p class='error'>{{ error }}</p>

{% endif %}

{% endblock %}

Login.html

{% extends 'base.html' %}

{% block content %}

<h2>Log In</h2>

<form action="/login" method='POST'/>

<p><label>Email<input type="email" name="email"/></label></p>

<p><label>Password<input type="password" name="password"/></label></p>

<p><input type="submit" value="log in"/></p>

</form>

{% endblock %}

rating-confirmation.html

{% extends "base.html" %}

{% block content %}

<p>

You gave

<strong>{{ movie.name }}</strong>

a rating of

<strong>{{ rating }}</strong>.

</p>

{% endblock %}

add-confirmation.html

{% extends "base.html" %}

{% block content %}

<p>

<strong>{{ movie.name }}</strong> has been added to your Watchlist.

</p>

{% endblock %}

crossoff.html

{% extends "base.html" %}

{% block content %}

<p><strike>{{ crossed\_off\_movie.name }}</strike> has been crossed off your Watchlist.</p>

{% endblock %}

Ratings.html

{% extends "base.html" %}

{% block content %}

<h2>Movies I Have Watched:</h2>

<ul>

{% for movie in movies %}

<li>

<h3>{{ movie.name }}</h3>

<form action="/rating-confirmation" method="post">

<label>

My Rating:

<select name="rating">

{% for rating in ["How was it?", "\*", "\*\*", "\*\*\*", "\*\*\*\*", "\*\*\*\*\*"] %}

<option {% if rating==movie.rating %}selected{% endif %}>{{ rating }}</option>

{% endfor %}

</select>

</label>

<input type="hidden" name="movie\_id" value="{{ movie.id }}"/>

<input type="submit" value="Rate it!"/>

</form>

</li>

{% endfor %}

</ul>

{% endblock %}

register.html

{% extends "base.html" %}

{% block content %}

<h2>Register New Account</h2>

<form action="/register" method="post">

<p><label>Email <input type="text" name="email"/></label></p>

<p><label>New Password <input type="password" name="password"/></label></p>

<p><label>Verify Password <input type="password" name="verify"/></label></p>

<p><input type="submit" value="Register"/></p>

</form>

{% endblock %}