Documentation

Introduction

This Flask application is a simple login system that allows users to register, log in, and log out. It uses a PostgreSQL database to store user information, and the Flask-SQLAlchemy extension to interact with the database.

Setup

Requirements:

- Make sure you have Python installed. You can download it from www.python.org.
- Create a virtual environment (optional but recommended).

Installation:

- Extract the provided zip file.
- Navigate to the extracted directory.
- Install the required dependencies using the following command:

```
pip install -r requirements.txt
```

Database Configuration:

- Install PostgreSQL and create a database named `ewsmetrics`.
  You can create the database with this command after login to psql shell with any user:

```
CREATE DATABASE EWSMetrics;
```

- Adjust the database URI in `app.py` to match your PostgreSQL credentials. Replace *username* and *password* with your username and password

Run the Application:

- Execute the following command to run the Flask application:

```
python app.py
```

- The application will be accessible at `http://127.0.0.1:5000/` in your web browser.

Code Explanation

Imports and Setu

```
from flask import Flask, url_for, render_template, request, redirect, session

from models import db, User
```

- Flask: A web framework for building web applications in Python.
- url_for: A function for generating URLs.
- render_template: Renders HTML templates.
- request: Handles incoming HTTP requests.
- redirect: Redirects the user to a different endpoint.
- session: A dictionary that stores user session variables.
- models: Imports the `db` instance and the `User` model from the `models.py` file.

Database Model

```
class User(db.Model):

 __tablename__ = 'app_users'



 id = db.Column(db.Integer, primary_key=True)

 full_name = db.Column(db.String(100))

 username = db.Column(db.String(100), unique=True)

 password = db.Column(db.String(100))



 def __init__(self, user
 self.full_name = full_name

 self.username = username

 self.password = password
```

- User Model: Represents the user table in the database.
- tablename: Sets the custom table name to 'app_users'.
- id: Primary key for the user table.
- full_name, username, password: Columns to store user information.
- init: Constructor to initialize a User object.

Flask App Configuration

```
app = Flask(__name__)

app.config['SECRET_KEY'] = 'dfvjh334gjhv$gjhjh'

app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://postgres:12345@localhost/ewsmetrics'

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

- Flask App Instance: Creates an instance of the Flask application.
- SECRET_KEY: Used for session management and should be kept secure.
- SQLALCHEMY_DATABASE_URI: Sets the PostgreSQL database URI.
- SQLALCHEMY_TRACK_MODIFICATIONS: Disables Flask-SQLAlchemy modification tracking.

Database Initialization

```
with app.app_context():

 db.create_all()
```

- Database Initialization: Creates all database tables within the application context.
- app.app_context(): Creates a context for the application.

Home Page Route

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

def index():

 if session.get('logged_in'):

 user = User.query.filter_by(username=session['username']).first().full_name

 return render_template('home.html', user=user)

 else:

 return render_template('index.html', message="Hello!")
```

- Home Page Route: Renders the home page.
- session.get('logged_in'): Checks if the user is logged in.
- User.query.filter_by(...).first().full_name: Retrieves the full name of the logged-in user.
- render_template(...): Renders the home or index template based on the user's login status.

User Registration Route

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

def register():

 if request.method == 'POST':

 try:
```

```
 new_user = User(full_name=request.form['full_name'],
username=request.form['username'], password=request.form['password'])

 db.session.add(new_user)

 db.session.commit()

 return redirect(url_for('login'))

 except:

 return render_template('register.html'          "User Already Exists"

 else:

 return render_template('register.html')
```

- User Registration Route: Handles user registration.
- request.method == 'POST': Checks if the form is submitted.
- User(...): Creates a new User object with registration form data.
- db.session.add(...): Adds the new user to the database.
- db.session.commit(): Commits the changes to the database.
- redirect(url_for(...)): Redirects to the login page after successful registration.

User Login Route

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

def login():

 if request.method == 'POST':

 u = request.form['username']

 p = request.form['password']
 data = User.query.filter_by(username=u, password=p).first()

 if data is not None:

 session['logged_in'] = True

 session['username'] = u

 return redirect(url_for('index'))

 return render_template('login.html', message="Incorrect Details")

 else:

 return render_template('login.html')
```

- User Login Route: Handles user login.
- request.method == 'POST': Checks if the form is submitted.
- User.query.filter_by(...).first(): Queries the database for the provided username and password.
- session['logged_in'] = True: Sets the 'logged_in' session variable to True upon successful login.
- redirect(url_for(...)): Redirects to the home page after successful login.

User Logout Route

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

def logout():

 session['logged_in'] = False

 session.pop('username', None)

 return redirect(url_for('index'))
```

- User Logout Route: Handles user logout.
- session['logged_in'] = False: Sets the 'logged_in' session variable to False.
- session.pop('username', None): Removes the 'username' from the session.

- redirect(url_for(...)): Redirects to the home page after successful logout.

`