**[Que-42] - How can you implement user authentication in a Flask application?**

Implementing user authentication in a Flask application involves creating routes for user registration, login, and logout, as well as protecting certain routes that require authentication. Here's a step-by-step guide to implementing user authentication using Flask, Flask-Login, and Flask-WTF.

### **Step 1: Install Required Packages**

First, install Flask, Flask-Login, Flask-WTF, and Flask-SQLAlchemy (for database management) using pip:

pip install flask flask-login flask-wtf flask-sqlalchemy

### **Step 2: Set Up the Flask Application**

Create a new directory for your project and within it, create the following files: app.py, models.py, forms.py, and a folder named templates for your HTML templates.

### **Step 3: Define the Flask Application**

**app.py**:

from flask import Flask, render\_template, redirect, url\_for, request, flash  
from flask\_sqlalchemy import SQLAlchemy  
from flask\_login import LoginManager, UserMixin, login\_user, logout\_user, login\_required, current\_user  
from werkzeug.security import generate\_password\_hash, check\_password\_hash  
from forms import LoginForm, RegisterForm  
  
app = Flask(\_\_name\_\_)  
app.config['SECRET\_KEY'] = 'your\_secret\_key'  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///users.db'  
db = SQLAlchemy(app)  
login\_manager = LoginManager(app)  
login\_manager.login\_view = 'login'  
  
# Define User model  
class User(UserMixin, db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 username = db.Column(db.String(150), unique=True, nullable=False)  
 password = db.Column(db.String(150), nullable=False)  
  
@login\_manager.user\_loader  
def load\_user(user\_id):  
 return User.query.get(int(user\_id))  
  
@app.route('/login', methods=['GET', 'POST'])  
def login():  
 form = LoginForm()  
 if form.validate\_on\_submit():  
 user = User.query.filter\_by(username=form.username.data).first()  
 if user and check\_password\_hash(user.password, form.password.data):  
 login\_user(user)  
 return redirect(url\_for('dashboard'))  
 flash('Invalid username or password')  
 return render\_template('login.html', form=form)  
  
@app.route('/register', methods=['GET', 'POST'])  
def register():  
 form = RegisterForm()  
 if form.validate\_on\_submit():  
 hashed\_password = generate\_password\_hash(form.password.data, method='sha256')  
 new\_user = User(username=form.username.data, password=hashed\_password)  
 db.session.add(new\_user)  
 db.session.commit()  
 return redirect(url\_for('login'))  
 return render\_template('register.html', form=form)  
  
@app.route('/dashboard')  
@login\_required  
def dashboard():  
 return f'Hello, {current\_user.username}! Welcome to your dashboard.'  
  
@app.route('/logout')  
@login\_required  
def logout():  
 logout\_user()  
 return redirect(url\_for('login'))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 db.create\_all()  
 app.run(debug=True)

### **Step 4: Define the User Model**

**models.py**:

from app import db  
  
class User(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 username = db.Column(db.String(150), unique=True, nullable=False)  
 password = db.Column(db.String(150), nullable=False)

### **Step 5: Create Forms for Login and Registration**

**forms.py**:

from flask\_wtf import FlaskForm  
from wtforms import StringField, PasswordField, SubmitField  
from wtforms.validators import DataRequired, Length, EqualTo  
  
class LoginForm(FlaskForm):  
 username = StringField('Username', validators=[DataRequired(), Length(min=4, max=150)])  
 password = PasswordField('Password', validators=[DataRequired()])  
 submit = SubmitField('Login')  
  
class RegisterForm(FlaskForm):  
 username = StringField('Username', validators=[DataRequired(), Length(min=4, max=150)])  
 password = PasswordField('Password', validators=[DataRequired()])  
 confirm\_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])  
 submit = SubmitField('Register')

### **Step 6: Create HTML Templates**

**templates/login.html**:

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Login</title>  
</head>  
<body>  
 <h1>Login</h1>  
 <form method="POST">  
 {{ form.hidden\_tag() }}  
 <p>  
 {{ form.username.label }}<br>  
 {{ form.username(size=32) }}  
 </p>  
 <p>  
 {{ form.password.label }}<br>  
 {{ form.password(size=32) }}  
 </p>  
 <p>{{ form.submit() }}</p>  
 </form>  
 <a href="{{ url\_for('register') }}">Register</a>  
</body>  
</html>

**templates/register.html**:

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Register</title>  
</head>  
<body>  
 <h1>Register</h1>  
 <form method="POST">  
 {{ form.hidden\_tag() }}  
 <p>  
 {{ form.username.label }}<br>  
 {{ form.username(size=32) }}  
 </p>  
 <p>  
 {{ form.password.label }}<br>  
 {{ form.password(size=32) }}  
 </p>  
 <p>  
 {{ form.confirm\_password.label }}<br>  
 {{ form.confirm\_password(size=32) }}  
 </p>  
 <p>{{ form.submit() }}</p>  
 </form>  
 <a href="{{ url\_for('login') }}">Login</a>  
</body>  
</html>

### **Step 7: Run the Application**

1. Open a terminal or command prompt.
2. Navigate to the directory where app.py is located.
3. Run the following command to start the Flask application:

python app.py

1. Open a web browser and go to <http://127.0.0.1:5000/>.

You should now have a basic Flask application with user authentication, including registration, login, and logout functionalities. The @login\_required decorator ensures that only authenticated users can access the dashboard route.