

Full Stack Experiment 8

Name: Abhay Mathur

SAPID: 60017210016

Batch: A1

Theory:

Abhay Mathur
60017210016

Full Stack Experiment 8

Aim: Handling user Input with forms in Flask

Theory: Flask Login provides a simplified way of managing users, including easily logging in and out users and restricting certain pages to authenticated users.

Step 1: Import necessary modules

- Import flask flask-sqlalchemy & flask-login
- Create your flask application, indicate what database flask-sqlalchemy should connect to & initialize the flask sqlalchemy extension.
- We initialize the login manager class from flask-login to be able to log in & out users

Step 2: Create a user Model & database

To be able to store user's info, we need to create a table with flask-sqlalchemy. This is done by creating a model that represents the info we want to store. We create user class a subclass of ~~user~~ UserMixin, which will help to implement properties such as ~~is~~ is_authenticated to the users class. We will also need to create columns we need to specify the database such as db.Integer & db.String as well. When creating columns we also need to specify keywords such as unique=True if we want to ensure values. Similarly we can set NULL, primary key field as well

Step 3: Adding a user Loader

Before implementing the functionality for authenticating the user, we need to specify a function that flask-login can use to retrieve a user object given a user id

Step 4: Registering New Accounts with flask login

- i) Add the following code to a file name sign up.html in a folder called templates. This will need to contain a form that allows the user to enter their details, such as their username & chosen password.
- ii) Create a route that renders the template & creates the user account if they make a ~~data~~ POST request.

Step 5: Allowing users to log in with Flask-login

- Like with creating the registered route we first need a way for users to login through an HTML form
- Add functionality to log in to user with a login function for the login route

Step 6: Conditionally ~~render~~ rendering HTML based on users authentication status with flask-login

- ~~data~~ We can use Jinja template to show user authentication status by using if else statements.

~~28/10/23~~

Code:

app.py:

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
```

```

from flask_login import LoginManager, UserMixin, login_user, logout_user

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///todo.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config["SECRET_KEY"] = "abc"
db=SQLAlchemy()

login_manager = LoginManager()
login_manager.init_app(app)

# Create a user model
class Users(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(250), nullable = False)
    password = db.Column(db.String(250), nullable = False)

# Initialize app with extension
db.init_app(app)

# Create database with context
with app.app_context():
    db.create_all()

@login_manager.user_loader
def loader_user(user_id):
    return Users.query.get(int(user_id))

@app.route('/register', methods=["GET", "POST"])
def register():
    # If the user made a POST request, create a new user
    if request.method == "POST":
        user = Users(username=request.form.get("username"),
                      password=request.form.get("password"))
        # Add the user to the database
        db.session.add(user)
        # Commit the changes made
        db.session.commit()
        # Once user account created, redirect them
        # to login route (created later on)
        return redirect(url_for("login"))
    # Renders sign_up template if user made a GET request
    return render_template("sign_up.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    # If a post request was made, find the user by
    # filtering for the username

```

```

if request.method == "POST":
    user = Users.query.filter_by(
        username=request.form.get("username")).first()
    # Check if the password entered is the
    # same as the user's password
    if user.password == request.form.get("password"):
        # Use the login_user method to log in the user
        login_user(user)
        return redirect(url_for("home"))
    # Redirect the user back to the home
    # (we'll create the home route in a moment)
    return render_template("login.html")

@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for("home"))

@app.route("/")
def home():
    # Render home.html on "/" route
    return render_template("home.html")

if __name__ == "__main__":
    app.run(debug=True)

```

home.html:

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Home</title>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>

<body>
    <nav>
        <ul>
            <li><a href="/login">Login</a></li>

```

```

        <li><a href="/register">Create account</a></li>
    </ul>
</nav>
{% if current_user.is_authenticated %}
<h1>You are logged in</h1>
<a href="/logout">Logout</a>
{% else %}
<h1>You are not logged in</h1>
{% endif %}
</body>
</html>

```

login.html:

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Login</title>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>

<body>
    <nav>
        <ul>
            <li><a href="/login">Login</a></li>
            <li><a href="/register">Create account</a></li>
        </ul>
    </nav>
    <h1>Login to your account</h1>
    <form action="#" method="post">
        <label for="username">Username:</label>
        <input type="text" name="username" />
        <label for="password">Password:</label>
        <input type="password" name="password" />
        <button type="submit">Submit</button>
    </form>
</body>
</html>

```

sign_up.html:

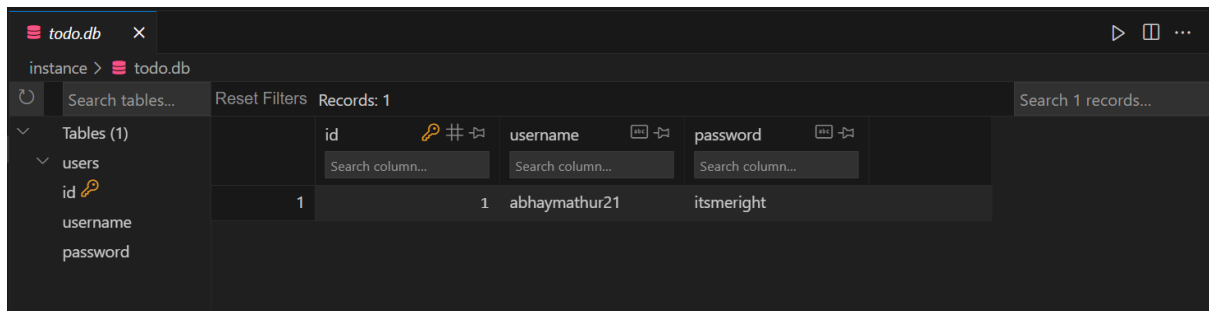
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Sign Up</title>
  <style>
    h1 {
      color: green;
    }
  </style>
</head>

<body>
  <nav>
    <ul>
      <li><a href="/login">Login</a></li>
      <li><a href="/register">Create account</a></li>
    </ul>
  </nav>
  <h1>Create an account</h1>
  <form action="#" method="post">
    <label for="username">Username:</label>
    <input type="text" name="username" />
    <label for="password">Password:</label>
    <input type="password" name="password" />
    <button type="submit">Submit</button>
  </form>
</body>

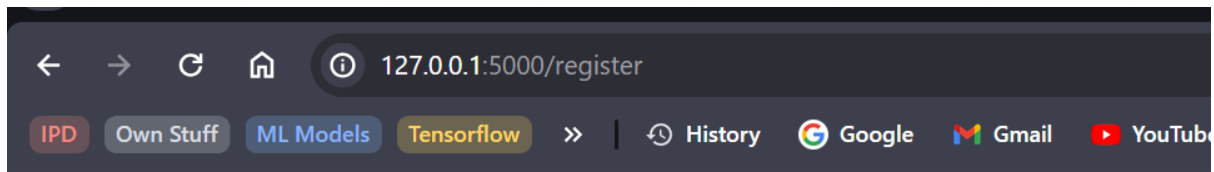
</html>
```

todo.db:



id	username	password
1	abhaymathur21	itsmeright

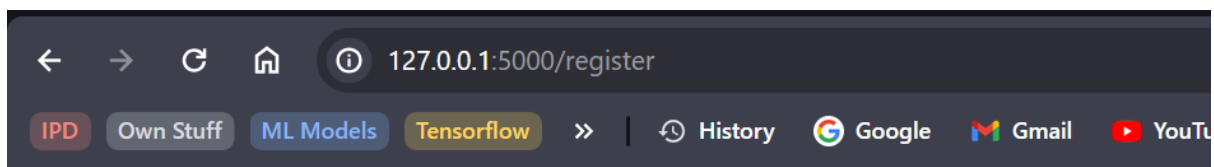
Output:



- [Login](#)
- [Create account](#)

Create an account

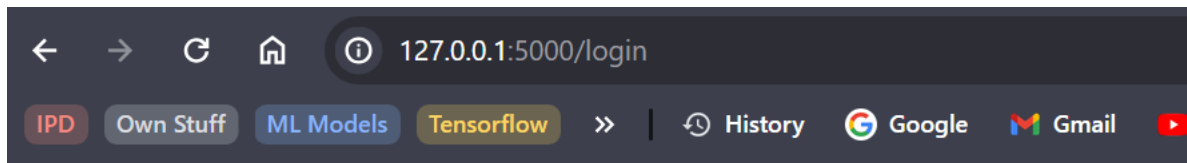
Username: Password:



- [Login](#)
- [Create account](#)

Create an account

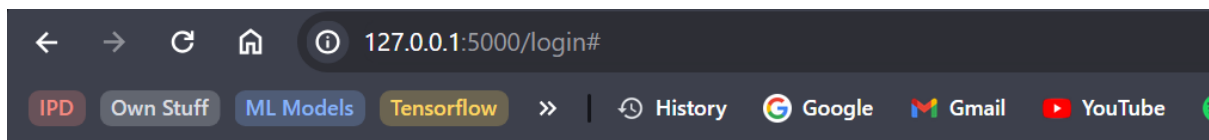
Username: Password:



- [Login](#)
- [Create account](#)

Login to your account

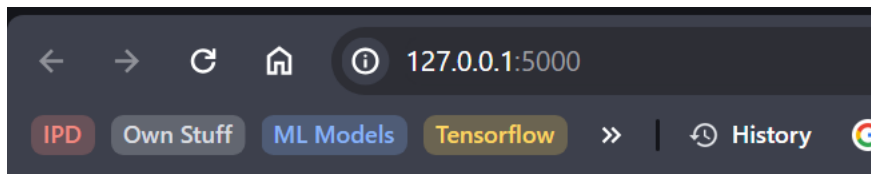
Username: Password:



- [Login](#)
- [Create account](#)

Login to your account

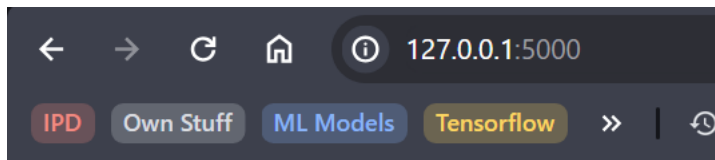
Username: Password:



- [Login](#)
- [Create account](#)

You are logged in

[Logout](#)



- [Login](#)
- [Create account](#)

You are not logged in

Conclusion: Learnt how to make a basic login page in flask and link it with an SQLite database.