How to store username and password in Flask

Last Updated : 20 Apr, 2023

We'll discuss how to save a Username and password on a Flask website in this article. We may view the welcome message and the username we chose when registering on the profile page after logging in with our username and password. The names of subsequent users who sign up using the login information we'll be using here will also show up on the profile page screen. To maintain the user login and registration credentials, Python code will be coupled to a Flask MySQL database. From there, we can utilize phpmyadmin to see how many users have registered and modified their information.

Required module

For making our project we install Flask, and Flask-MySQLdb, and create a virtual environment.

```
pip install Flask-MySQLdb
pip install Flask
```

Templates Files

Create a template folder and in the templates folder we basically made three files one for register.html, another one for login.html, and at last one for the user.html.

register.html

This HTML file contains a straightforward registration form that asks for three inputs: username, email address, and password. Once these fields have been completed, click the register button to see a flashing message stating that the form has been successfully submitted and that the registration information has been safely saved in the MySQL database. For flashing massage, we are using

Jinja2 in an HTML file, so we can now log in using our credentials. If we registered using the same email address, the flash email id will also exist.

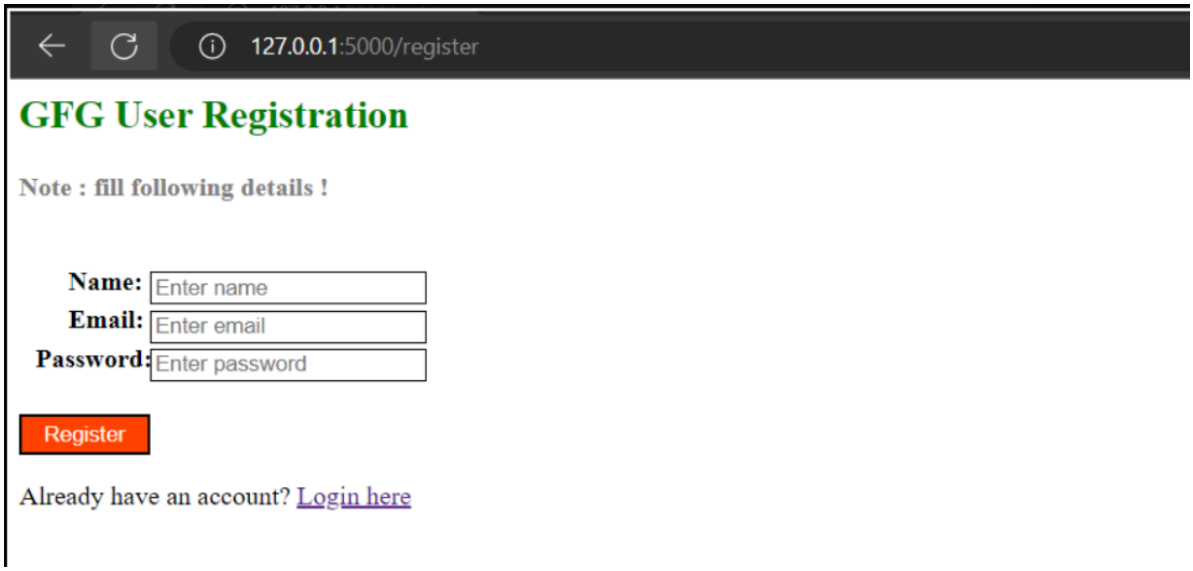
HTML

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Registration Form</title>

</head>
<style>
    .hi{
        color: green;
    }
    .ok{
        display: block;
        margin-left: 80px;
        margin-top: -15px;
        border: 1px solid black;
    }
    .gfg{
        margin-left: 30px;
        font-weight: bold;
    }
    .gf{
        margin-left: 10px;
        font-weight: bold;
    }
    .btn{
        margin-top: 20px;
        width: 80px;
        height: 25px;
        background-color: orangered;
        color: white;
    }
    .y{
        color: gray;
    }
</style>
<body>
<div class="container">
    <h2 class="hi" > GFG User Registration </h2>
    <h4 class="y"  >Note : fill following details !</h4>
    <form action="{{ url_for('register') }}" method="post">
        {% if message is defined and message %}
            <div class="alert alert-warning">  <strong>  {{ message }} ???? </str
```

```
        {% endif %}
        <br>
        <div class="form-group">
            <label class="gfg">Name:</label>
             <input class="ok" type="text" class="form-control" id="name" name="r
        </div>
        <div class="form-group">
            <label class="gfg">Email:</label>
             <input class="ok" type="email" class="form-control" id="email" name=
        </div>
        <div class="form-group">
            <label class="gf">Password:</label>
        <input class="ok" type="password" class="form-control" id="password" name
        </div>
        <button class="btn" type="submit" class="btn btn-primary">Register</butto
        <p class="bottom">Already have an account?  <a class="bottom" href="{{url
    </form>
</div>
</body>
</html>
```

Output:



register.html

login.html

In login.html, we have created two straightforward inputs: a username and a password that we successfully registered. If we enter the correct email address and password, It will direct us to the user/login profile page, where we have

used the URL for the function to write the file function that we want to display following successful registration.

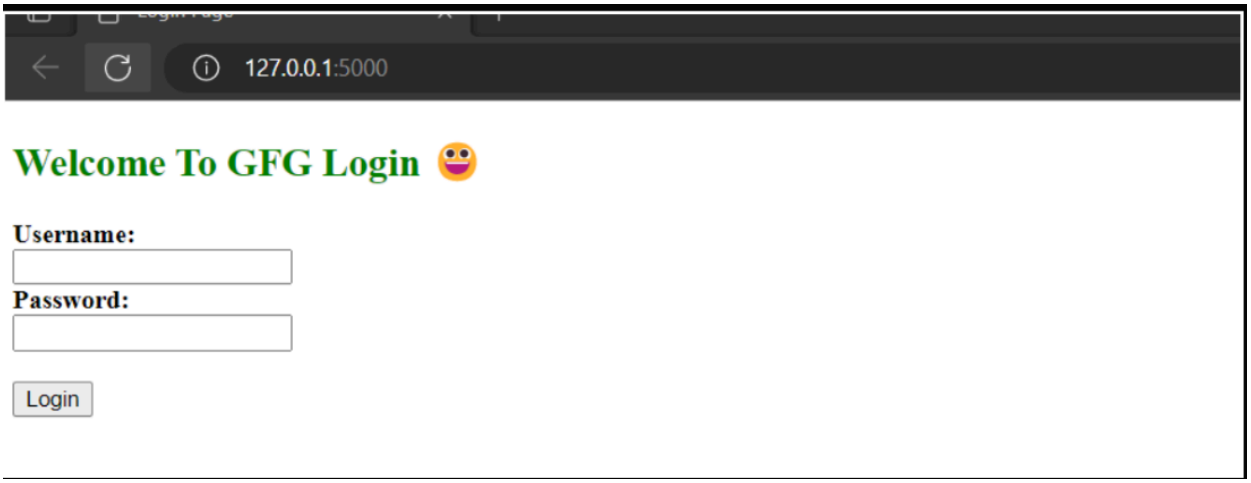
HTML

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Registration Form</title>

</head>
<style>
    .hi{
        color: green;
    }
    .ok{
        display: block;
        margin-left: 80px;
        margin-top: -15px;
        border: 1px solid black;
    }
    .gfg{
        margin-left: 30px;
        font-weight: bold;
    }
    .gf{
        margin-left: 10px;
        font-weight: bold;
    }
    .btn{
        margin-top: 20px;
        width: 80px;
        height: 25px;
        background-color: orangered;
        color: white;
    }
    .y{
        color: gray;
    }
</style>
<body>
<div class="container">
    <h2 class="hi" > GFG User Registration </h2>
    <h4 class="y"  >Note : fill following details !</h4>
    <form action="{{ url_for('register') }}" method="post">
        {% if message is defined and message %}
            <div class="alert alert-warning">  <strong>  {{ message }} ???? </str
```

```
        {% endif %}
        <br>
        <div class="form-group">
            <label class="gfg">Name:</label>
             <input class="ok" type="text" class="form-control" id="name" name="r
        </div>
        <div class="form-group">
            <label class="gfg">Email:</label>
             <input class="ok" type="email" class="form-control" id="email" name=
        </div>
        <div class="form-group">
            <label class="gf">Password:</label>
        <input class="ok" type="password" class="form-control" id="password" name
        </div>
        <button class="btn" type="submit" class="btn btn-primary">Register</butto
        <p class="bottom">Already have an account?  <a class="bottom" href="{{url
    </form>
</div>
</body>
</html>
```

Output:



login.html

user.html

After a successful login, we put a few lines of code to greet the user in these files. We also add a second session. The name code we use during registration means that when we log in, our name will also appear on the screen. Additionally, a button for logging out will appear on the screen; by clicking this button, we can log out and must log in again.

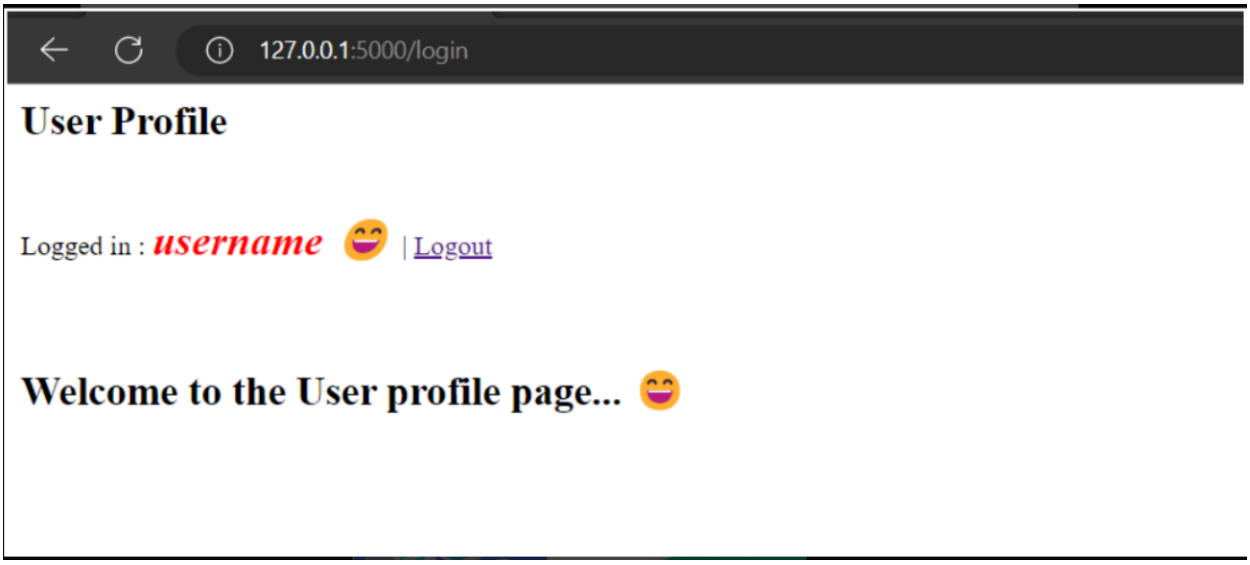
HTML

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Account</title>

</head>
<style>
.gfg{
    font-size: 25px;
    color: red;
    font-style: italic;
}
</style>
<body>
<div class="container">
    <div class="row">
        <h2>User Profile</h2>
    </div>
    <br>
    <div class="row">
        Logged in : <strong class="gfg">  {{session.name}} ???? </strong>| <a hre
    </div>
    <br><br>
    <div class="row">

        <h2>Welcome to the User profile page... ????</h2>
    </div>
</div>
</body>
</html>
```

Output:



user.html

Setting up app.py file

Step 1: Import all library

First, the python code is written in a file called app.py. We import all the libraries required for running our application, connecting to MySQL, and performing admin login from the database into this file. Following the import of re(regular expression), which will read the data from our MySQL database, the Python code database, MySQL DB, is used to construct the data for our database. We initialize the flask function and generate a secret key for our flask after importing all modules. The database name, email address, and password are then added to the database.

Python3

```
# Import all important libraries
from flask import *
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re

# initialize first flask
app = Flask(__name__)
app.secret_key = 'GeeksForGeeks'

# Set MySQL data
app.config['MYSQL_HOST'] = 'localhost'
```

```
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'user-table'

mysql = MySQL(app)
```

Step 2: Adding login and logout function

Then, we develop a functional login() and develop a session for login and registration for a system that also obtains our data from MySQL. A successfully registered message will also appear on the login page when we successfully register on the register page. In this function, we pass the request to the login form by entering our name, password, and email when we click on enter. It will automatically save on our PHPMyAdmin by MySQL data.

HTML

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Registration Form</title>

</head>
<style>
    .hi{
        color: green;
    }
    .ok{
        display: block;
        margin-left: 80px;
        margin-top: -15px;
        border: 1px solid black;
    }
    .gfg{
        margin-left: 30px;
        font-weight: bold;
    }
    .gf{
        margin-left: 10px;
        font-weight: bold;
    }
    .btn{
        margin-top: 20px;
```

```
            width: 80px;
            height: 25px;
            background-color: orangered;
            color: white;
        }
        .y{
            color: gray;
        }
    </style>
    <body>
    <div class="container">
        <h2 class="hi" > GFG User Registration </h2>
        <h4 class="y"  >Note : fill following details !</h4>
        <form action="{{ url_for('register') }}" method="post">
            {% if message is defined and message %}
                <div class="alert alert-warning">  <strong>  {{ message }} ???? </str
            {% endif %}
            <br>
            <div class="form-group">
                <label class="gfg">Name:</label>
                 <input class="ok" type="text" class="form-control" id="name" name="r
            </div>
            <div class="form-group">
                <label class="gfg">Email:</label>
                 <input class="ok" type="email" class="form-control" id="email" name=
            </div>
            <div class="form-group">
                <label class="gf">Password:</label>
            <input class="ok" type="password" class="form-control" id="password" name
            </div>
            <button class="btn" type="submit" class="btn btn-primary">Register</butto
            <p class="bottom">Already have an account?  <a class="bottom" href="{{url
        </form>
    </div>
    </body>
    </html>
```

Step 3: Adding User Registration function

On the login screen, we can log in by providing our email address and password. There are also more flashing notifications, such as “User already exists” if we attempt to register again using the same email address. With the same email address, we are able to create two registered accounts. The username we specified on the registration page will also show up on the

profile once we successfully log in. For this instance, we typed “GFG.” “Welcome GFG” will appear once we have successfully logged in. That clarifies the entire code as well as its intended use.

Python3

```
# Make a register session for registration
# session and also connect to Mysql to code for access
# login and for completing our login
# session and making some flashing massage for error
@app.route('/register', methods=['GET', 'POST'])
def register():
    message = ''
    if request.method == 'POST' and 'name' in
        request.form and 'password' in request.form
            and 'email' in request.form:

        userName = request.form['name']
        password = request.form['password']
        email = request.form['email']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM user WHERE email = % s',
                       (email, ))
        account = cursor.fetchone()
        if account:
            message = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            message = 'Invalid email address !'
        elif not userName or not password or not email:
            message = 'Please fill out the form !'
        else:
            cursor.execute(
                'INSERT INTO user VALUES (NULL, % s, % s, % s)',
              (userName, email, password, ))
            mysql.connection.commit()
            message = 'You have successfully registered !'
    elif request.method == 'POST':
        message = 'Please fill out the form !'
    return render_template('register.html', message=message)
```

Complete Code

Python3

```
# Import all important libraries
from flask import *
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re

# initialize first flask
app = Flask(__name__)
app.secret_key = 'GeeksForGeeks'

# Set MySQL data
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'user-table'

mysql = MySQL(app)


@app.route('/')
@app.route('/login', methods=['GET', 'POST'])
def login():
    message = ''
    if request.method == 'POST' and 'email' in
    request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor
                (MySQLdb.cursors.DictCursor)
        cursor.execute(
            'SELECT * FROM user WHERE email = % s AND password = % s',
                 (email, password, ))
        user = cursor.fetchone()
        if user:
            session['loggedin'] = True
            session['userid'] = user['userid']
            session['name'] = user['name']
            session['email'] = user['email']
            message = 'Logged in successfully !'
            return render_template('user.html',
                                   message=message)
        else:
            message = 'Please enter correct email / password !'
    return render_template('login.html',
                           message=message)
```

```
# Make function for logout session
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('userid', None)
    session.pop('email', None)
    return redirect(url_for('login'))


@app.route('/register', methods=['GET', 'POST'])
def register():
    message = ''
    if request.method == 'POST' and 'name' in request.form
            and 'password' in request.form and 'email' in request.form:
        userName = request.form['name']
        password = request.form['password']
        email = request.form['email']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM user WHERE email = % s', (email, ))
        account = cursor.fetchone()
        if account:
            message = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            message = 'Invalid email address !'
        elif not userName or not password or not email:
            message = 'Please fill out the form !'
        else:
            cursor.execute(
                'INSERT INTO user VALUES (NULL, % s, % s, % s)',
                     (userName, email, password ))
            mysql.connection.commit()
            message = 'You have successfully registered !'
    elif request.method == 'POST':
        message = 'Please fill out the form !'
    return render_template('register.html', message=message)


# run code in debug mode
if __name__ == "__main__":
    app.run(debug=True)
```

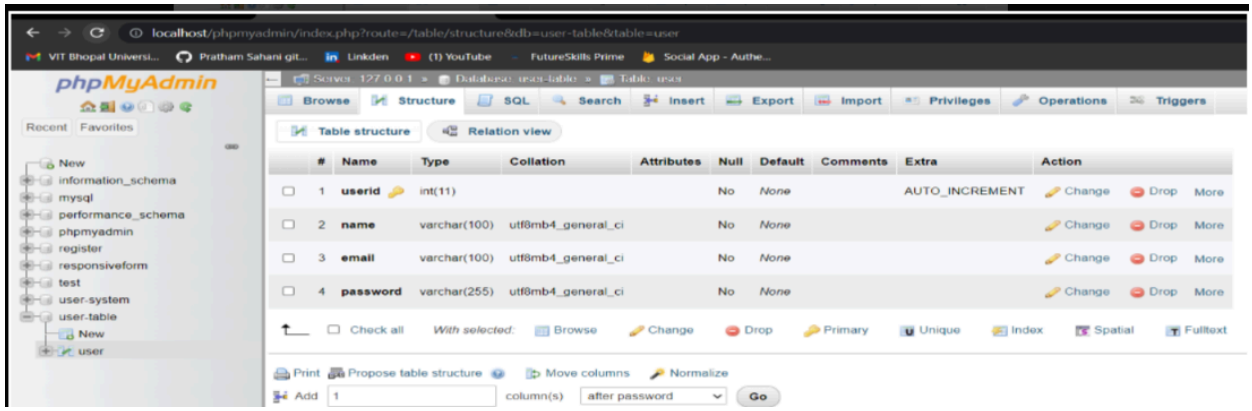
After writing whole open your terminal and run the following command

```
python app.py
```

Output:

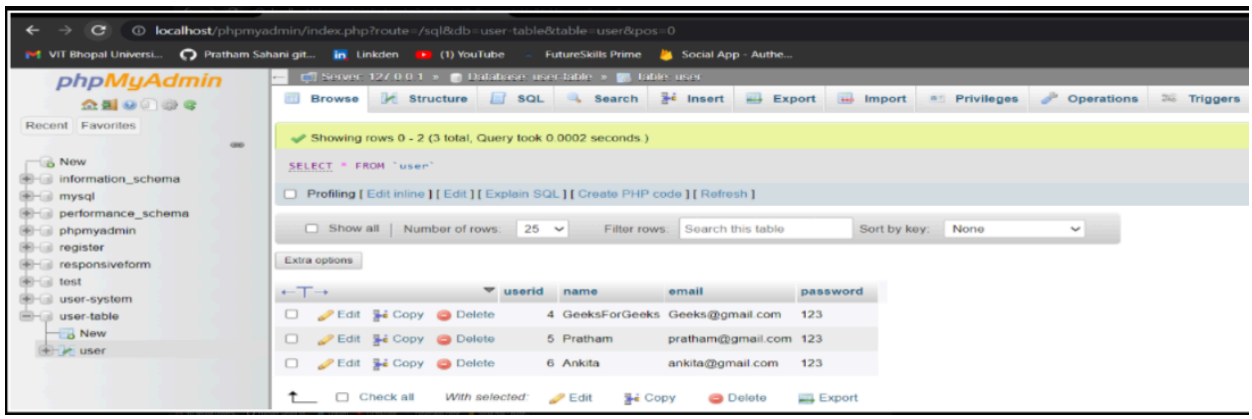
Complete the output and showing where we save our password and username and how.

Database Output:



Database table

After registering multiple users these outputs will show in your database by watching the video you can understand how the username will display on the screen and how multiple users can register and log in.



Save/Store username and password

3

Previous Article

Password Hashing with Bcrypt in Flask

Next Article

Flask - Role Based Access Control

Similar Reads

Add User and Display Current Username in Flask

In this article, we'll talk about how to add a User and Display the Current Username on a Flask website. When we log in using our username and...

9 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application.The Session is the time between the client logs in to...

4 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

Password Hashing with Bcrypt in Flask

In this article, we will use Password Hashing with Bcrypt in Flask using Python. Password hashing is the process of converting a plaintext password into a...

2 min read

Python PRAW – Getting the username of a redditor

In Reddit, a redditor is the term given to a user. Each and every redditor has a username chosen by them during registration. Here we will see how to fetch the...

2 min read

How to generate a unique username using Python

A username is a unique name that is mainly used on websites and apps. A username is used to identify a person. For example, if your name is Akshay and...

4 min read

How to get the current username in Python

When building interacting Python programs you might need to get the current username for features like personalizing user experience, providing user-specific...

4 min read

Article Tags : Python

Practice Tags : python

Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305