



Login and Registration Project Using Flask and MySQL

Last Updated : 09 Aug, 2022

Project Title: Login and registration Project using Flask framework and MySQL Workbench. **Type of Application (Category):** Web application. **Introduction:** A framework is a code library that makes a developer's life easier when building web applications by providing reusable code for common operations. There are a number of frameworks for Python, including Flask, Tornado, Pyramid, and Django. Flask is a lightweight web application framework. It is classified as a micro-framework because it does not require particular tools or libraries. **Pre-requisite:** Knowledge of Python, MySQL Workbench and basics of Flask Framework. Python and MySQL Workbench should be installed in the system. Visual studio code or Spyder or any code editor to work on the application. **Technologies used in the project:** Flask framework, MySQL Workbench. **Implementation of the Project: (1) Creating Environment Step-1:** Create an environment. Create a project folder and a venv folder within.

```
py -3 -m venv venv
```

[Flask Templates](#) [Jinja2](#) [Flask-REST API](#) [Python SQLAlchemy](#) [Flask Bcrypt](#) [Flask Cookies](#) [Json](#) [Postman](#)

```
venv\Scripts\activate
```

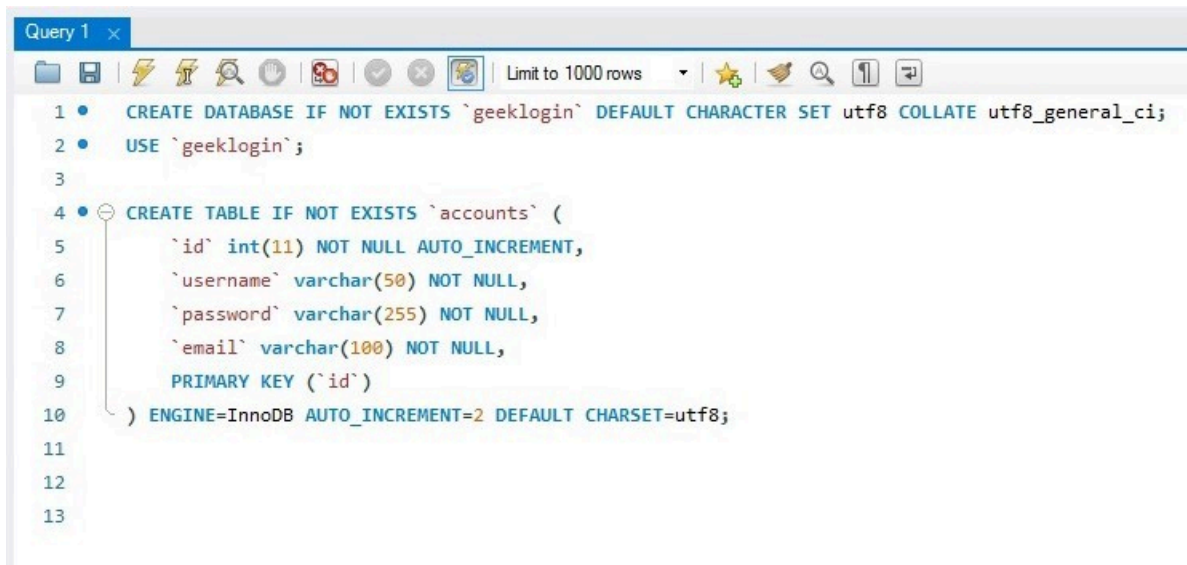
Step-3: Install Flask.

```
pip install Flask
```

(2) MySQL Workbench Step-1: Install MySQL workbench. Link to install : <https://dev.mysql.com/downloads/workbench/> Know more about it : <https://www.mysql.com/products/workbench/> **Step-2:** Install 'mysqlbd' module in your venv.

```
pip install flask-mysqldb
```

Step-3: Open MySQL workbench. **Step-4:** Write the following code. The above SQL statement will create our database **geeklogin** with the table **accounts**. **Step-5:** Execute the query.



```

Query 1 x
1 • CREATE DATABASE IF NOT EXISTS `geeklogin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
2 • USE `geeklogin`;
3
4 • CREATE TABLE IF NOT EXISTS `accounts` (
5     `id` int(11) NOT NULL AUTO_INCREMENT,
6     `username` varchar(50) NOT NULL,
7     `password` varchar(255) NOT NULL,
8     `email` varchar(100) NOT NULL,
9     PRIMARY KEY (`id`)
10 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
11
12
13

```

(3) Creating Project Step-1: Create an empty folder 'login'. **Step-2:** Now open your code editor and open this 'login' folder. **Step-3:** Create 'app.py' folder and write the code given below.

Python3

Store this code in 'app.py' file

```

from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re

```

```
app = Flask(__name__)
```

```

app.secret_key = 'your secret key'

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'your password'
app.config['MYSQL_DB'] = 'geeklogin'

mysql = MySQL(app)

@app.route('/')
@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE username = % s AND password = % s')
        account = cursor.fetchone()
        if account:
            session['loggedin'] = True
            session['id'] = account['id']
            session['username'] = account['username']
            msg = 'Logged in successfully !'
            return render_template('index.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE username = % s', (username, ))
        account = cursor.fetchone()
        if account:
            msg = 'Account already exists !'

```

```

elif not re.match(r'^@+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'Username must contain only characters and numbers !'
elif not username or not password or not email:
    msg = 'Please fill out the form !'
else:
    cursor.execute('INSERT INTO accounts VALUES (NULL, % s, % s, % s)', (u
mysql.connection.commit()
    msg = 'You have successfully registered !'
elif request.method == 'POST':
    msg = 'Please fill out the form !'
return render_template('register.html', msg = msg)

```

Step-4: Create the folder ‘**templates**’. create the file ‘login.html’, ‘register.html’, ‘index.html’ inside the ‘templates’ folder. **Step-5:** Open ‘login.html’ file and write the code given below. In ‘login.html’, we have two fields i.e. username and password. When user enters correct username and password, it will route you to index page otherwise ‘Incorrect username/password’ is displayed.

html

<!-- Store this code in 'login.html' file inside the 'templates' folder -->

```

<html>
  <head>
    <meta charset="UTF-8">
    <title> Login </title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
  </head>
  <body><br><br><br><br><br>
    <div align="center">
      <div align="center" class="border">
        <div class="header">
          <h1 class="word">Login</h1>
        </div><br><br><br>
        <h2 class="word">
          <form action="{{ url_for('login') }}" method="post">
            <div class="msg">{{ msg }}</div>
            <input id="username" name="username" type="text" placeholder="Username" />
            <input id="password" name="password" type="password" placeholder="Password" />
            <input type="submit" class="btn" value="Sign In"><br><br>
          </form>
        </h2>
      </div>
    </div>
  </body>
</html>

```

```

        <p class="bottom">Don't have an account? <a class="bottom" href="{{u
    </div>
</div>
</body>
</html>

```

Step-6: Open 'register.html' file and write the code given below. In 'register.html', we have three fields i.e. username, password and email. When user enters all the information, it stored the data in the database and 'Registration successful' is displayed.

html

<!-- Store this code in 'register.html' file inside the 'templates' folder -->

```

<html>
  <head>
    <meta charset="UTF-8">
    <title> Register </title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
  </head>
  <body><br><br><br><br><br>
    <div align="center">
      <div align="center" class="border">
        <div class="header">
          <h1 class="word">Register</h1>
        </div><br><br><br>
        <h2 class="word">
          <form action="{{ url_for('register') }}" method="post">
            <div class="msg">{{ msg }}</div>
            <input id="username" name="username" type="text" placeholder="Username" />
            <input id="password" name="password" type="password" placeholder="Password" />
            <input id="email" name="email" type="text" placeholder="Enter Email" />
            <input type="submit" class="btn" value="Sign Up"><br>
          </form>
        </h2>
        <p class="bottom">Already have an account? <a class="bottom" href="{{u
      </div>
    </div>
  </body>
</html>

```

Step-7: Open 'index.html' file and write the code given below. This page is displayed when login is successful and username is also displayed. The logout functionality is also included in this page. When user logs out, it moves to fresh login page again.

html

```
<!-- Store this code in 'index.html' file inside the 'templates' folder-->

<html>
  <head>
    <meta charset="UTF-8">
    <title> Index </title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body><br><br><br><br><br>
    <div align="center">
      <div align="center" class="border">
        <div class="header">
          <h1 class="word">Index</h1>
        </div><br><br><br>
        <div class="bottom">
          Hi {{session.username}}!!<br><br> Welcome to the index page
        </div><br><br><br>
        <a href="{{ url_for('logout') }}" class="btn">Logout</a>
      </div>
    </div>
  </body>
</html>
```

Step-8: Create the folder 'static'. create the file 'style.css' inside the 'static' folder and paste the given CSS code.

CSS

```
/* Store this code in 'style.css' file inside the 'static' folder*/

.header{
  padding: 5px 120px;
```

```
width: 150px;
height: 70px;
background-color: #236B8E;
}

.border{
padding: 80px 50px;
width: 400px;
height: 450px;
border: 1px solid #236B8E;
border-radius: 0px;
background-color: #9AC0CD;
}

.btn {
padding: 10px 40px;
background-color: #236B8E;
color: #FFFFFF;
font-style: oblique;
font-weight: bold;
border-radius: 10px;
}

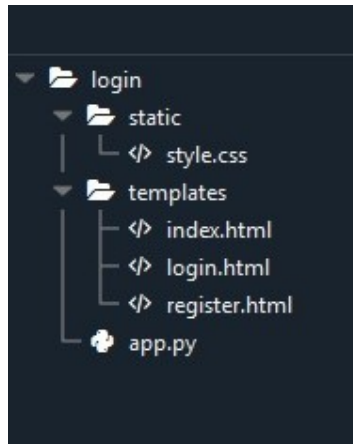
.textbox{
padding: 10px 40px;
background-color: #236B8E;
text-color: #FFFFFF;
border-radius: 10px;
}

::placeholder {
color: #FFFFFF;
opacity: 1;
font-style: oblique;
font-weight: bold;
}

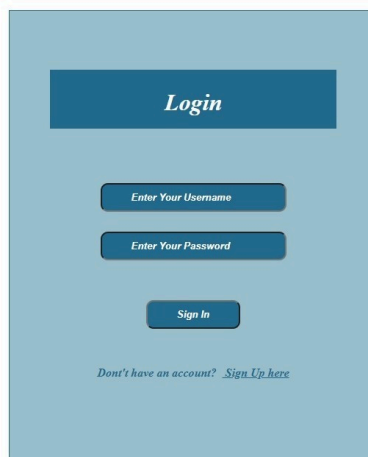
.word{
color: #FFFFFF;
font-style: oblique;
font-weight: bold;
}

.bottom{
color: #236B8E;
font-style: oblique;
font-weight: bold;
}
```

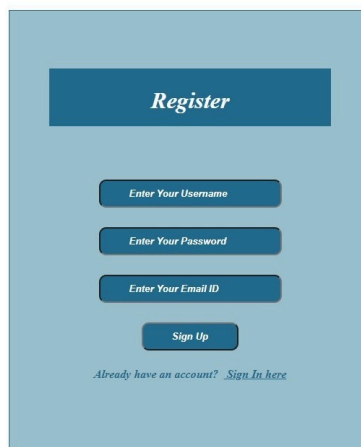
Step-9: The project structure will look like this.



(4) Run the Project Step-1: Run the server. **Step-2:** Browse the URL 'localhost:5000'. **Step-3:** The output web page will be displayed. **(5) Testing of the Application Step-1:** If you are new user, go to sign up page and fill the details. **Step-2:** After registration, go to login page. Enter your username and password and sign in. **Step-3:** If your login is successful, you will be moved to index page and your name will be displayed. **Output: Login page:**

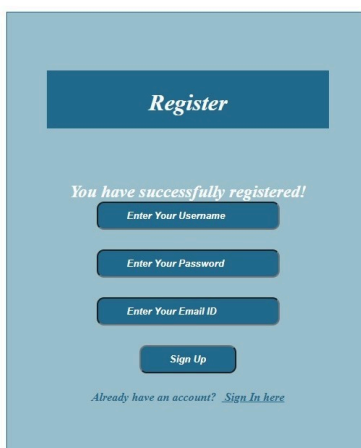


Registration page:



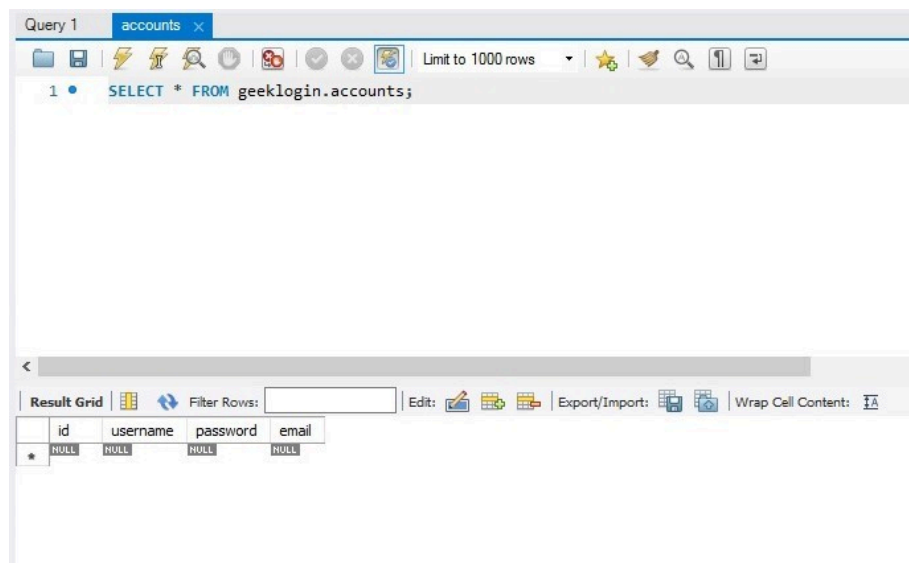
A registration form with a light blue background. At the top is a dark blue button labeled "Register". Below it are three input fields with dark blue borders and labels: "Enter Your Username", "Enter Your Password", and "Enter Your Email ID". Below these is a dark blue "Sign Up" button. At the bottom, there is a link that says "Already have an account? Sign In here".

If registration successful:



The same registration form as above, but with a message "You have successfully registered!" displayed in the center. The "Sign Up" button is still present, and the "Already have an account? Sign In here" link is at the bottom.

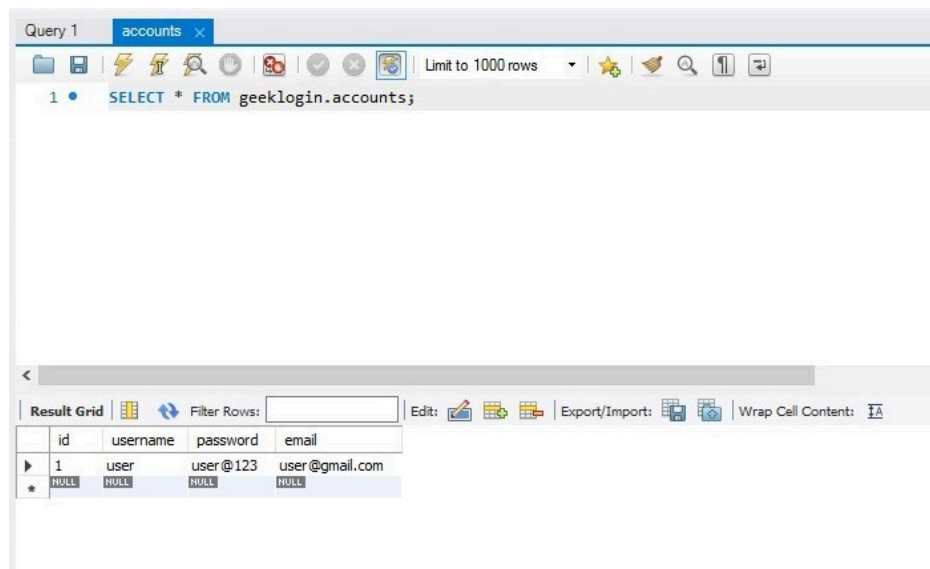
Before registration, Database table:



A screenshot of a database management tool showing a query result. The query is "SELECT * FROM geeklogin.accounts;". The result grid shows a single row with all NULL values for the columns id, username, password, and email.

id	username	password	email
*	NULL	NULL	NULL

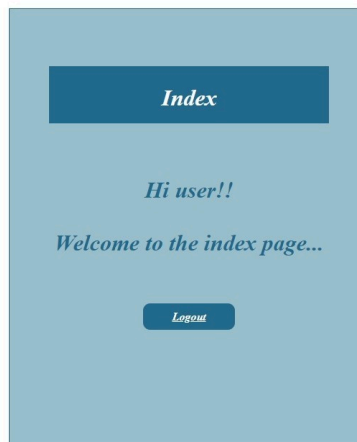
After registration, Database table:



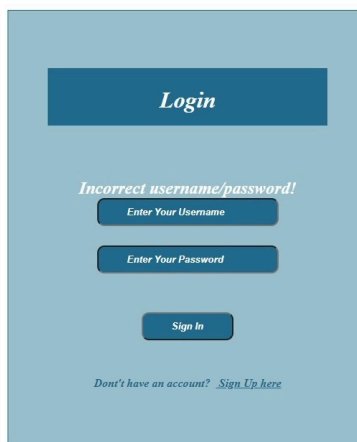
The screenshot shows a MySQL query client window titled 'Query 1' with a tab for 'accounts'. The query executed is 'SELECT * FROM geeklogin.accounts;'. The result is displayed in a 'Result Grid' with columns 'id', 'username', 'password', and 'email'. The first row shows '1', 'user', 'user@123', and 'user@gmail.com'. Below the first row, there are four 'NULL' values, likely representing the result of a second query or a different state.

	id	username	password	email
▶	1	user	user@123	user@gmail.com
★	NULL	NULL	NULL	NULL

If login successful, Indexpage is displayed:



If Login fails:



Looking to dive into the world of programming or sharpen your Python skills? Our [Master Python: Complete Beginner to Advanced Course](#) is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!



venni...



27

Previous Article

Making a Flask app using a PostgreSQL database

Next Article

How to execute raw SQL in Flask-SQLAlchemy app

Similar Reads

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

Creating a Registration and Login System with PHP and MySQL

A registration and login system is a fundamental component of many web applications and provides user authentication and security. This allows users to...

13 min read

How to make responsive sliding login and registration form using HTML CS...

In this article, we will learn how to make responsive sliding login and registration forms using HTML, CSS, and javascript. Forms are used on webpages for the us...

4 min read

How to Implement Email Registration and Login using Firebase in React?

Implementing Email Registration and Login using Firebase in React allows us to authenticate the users securely. Firebase provides predefined ways to register...

4 min read

Basic Registration and Login Form Using React Hook Form

In ReactJS, creating a form can be a nightmare as it involves handling inputs and validating inputs. React-hook-form is a ReactJS library that simplifies the proces...

6 min read

Design a Responsive Sliding Login & Registration Form using HTML CSS &...

In this article, we will see how to create the Responsive Sliding Login & Registration Form using HTML CSS & JavaScript, along with understanding its...

5 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

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 login without database - Python

In this article, we will talk about Python-based Flask login without a database in this article. In order to use Flask login without a database in this method basical...

4 min read

How To Add Authentication to Your App with Flask-Login

Whether it is building a simple blog or a social media site, ensuring user sessions are working correctly can be tricky. Fortunately, Flask-Login provides a simplifie...

9 min read

Article Tags :

PythonWeb TechnologiesFlask ProjectsPython Flask

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



Company

- About Us
- Legal
- In Media
- Contact Us
- Advertise with us
- GFG Corporate Solution
- Placement Training Program
- GeeksforGeeks Community

DSA

- Data Structures
- Algorithms
- DSA for Beginners
- Basic DSA Problems
- DSA Roadmap
- Top 100 DSA Interview Problems
- DSA Roadmap by Sandeep Jain
- All Cheat Sheets

Languages

- Python
- Java
- C++
- PHP
- GoLang
- SQL
- R Language
- Android Tutorial
- Tutorials Archive

Data Science & ML

- Data Science With Python
- Data Science For Beginner
- Machine Learning
- ML Maths
- Data Visualisation
- Pandas
- NumPy
- NLP
- Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Interview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved