

## Whom?

I have some friends who work in the exchange market. One of the hard problems is sharing information about forex exchange. People can use this website to share forex trading information confidentially. Using this website has the following advantages.

- Share information quickly without delay.
- Access is only available to authorized access.

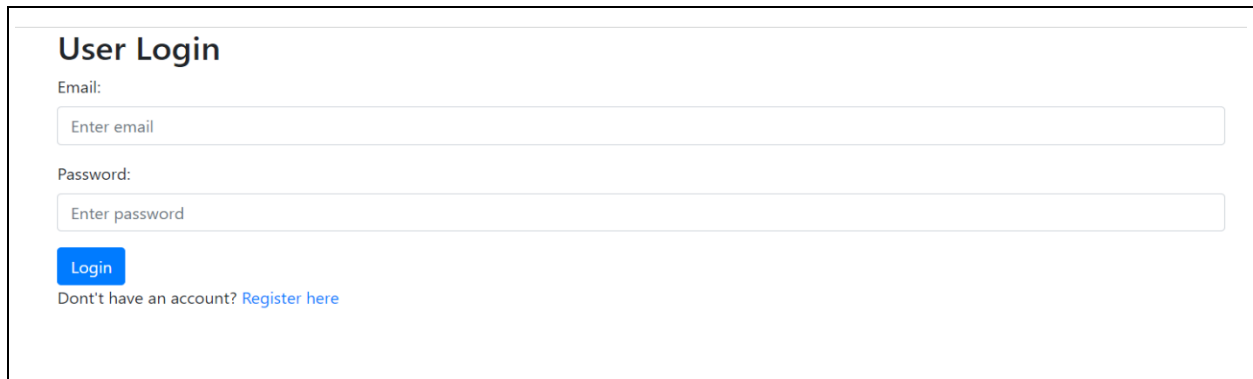
## What?

I used python flask and SQLiteStudio. This website has 3 pages (user login, user registration, and user profile) and it is connected to SQLiteStudio database.

### ➤ User login page:

You can first see a **user login** page when you visit the website. If you have access, you can visit the **user profile** page by entering your registered email address and password and clicking the **login** button. But if you don't have access, you need to click on **register here**.

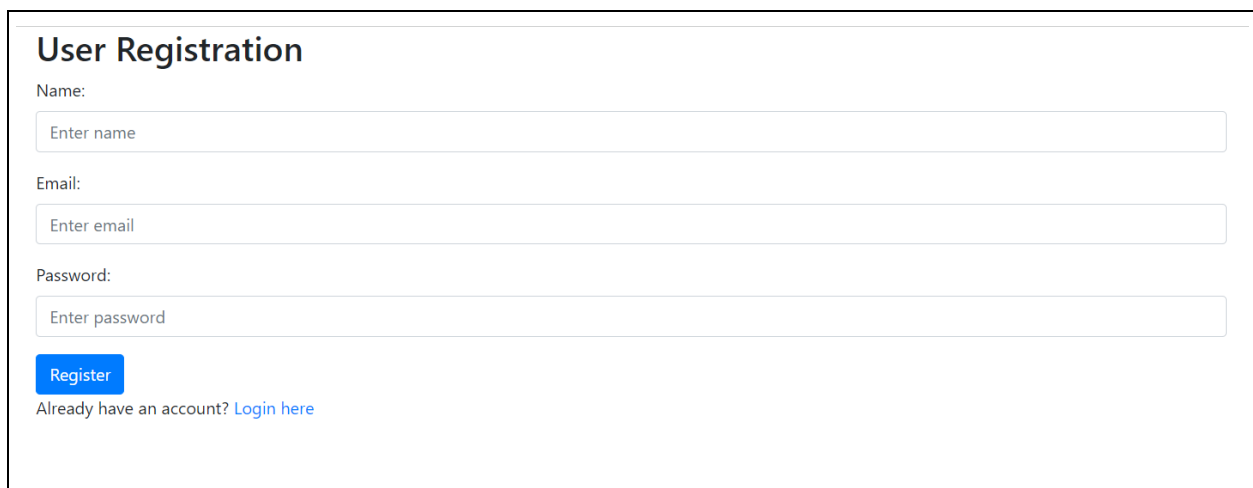
If you enter the wrong email or password, you will show "Please enter correct email / password !" message. If you enter the correct email or password, you will show "Logged in successfully !" message.



The image shows a web form titled "User Login". It contains two input fields: "Email:" with a placeholder "Enter email" and "Password:" with a placeholder "Enter password". Below the password field is a blue button labeled "Login". At the bottom, there is a link that says "Don't have an account? [Register here](#)".

### ➤ User registration page:

If you don't have access, you need to fill in your name, email, and password. After it, you must click on the **Register** button. After it, your data will be added to the **user1** table of the database. If your data is saved successfully, you will show "You have successfully registered !" message. After that, go to the login page and enter your registered email and password to visit the user profile page.



The image shows a web form titled "User Registration". It contains three input fields: "Name:" with a placeholder "Enter name", "Email:" with a placeholder "Enter email", and "Password:" with a placeholder "Enter password". Below the password field is a blue button labeled "Register". At the bottom, there is a link that says "Already have an account? [Login here](#)".

➤ User profile page

After visiting the **user profile** page, you will see the name of the user who is logged in. There is also a **log-out** button and clicking on it, it will take you to the **login** page. This page has three sections.

Section 1: After login, you will see a table of all requests for currency exchange. This table is saved as exchange1 in the database. You can use the information in this table to exchange the currency you need in a short time. Description of data in the table of all requests for currency exchange.

- Name – the name of the person making the request.
- Date – date of request
- Count – the amount of currency
- Exch – which currency exchange
- Type – buy or sell currency
- Phone – contact phone number

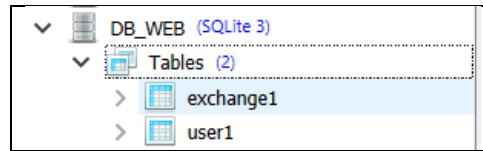
Section 2: from this table, you can see only your requests.

Section 3: In this section, you can enter a new request or delete an old request. To do this, you need to enter information such as Name, Date, Count, Currency, Type, and Phone. If you want to enter a new request, click the **save** button. But if you want to delete an old request, click the **delete** button.

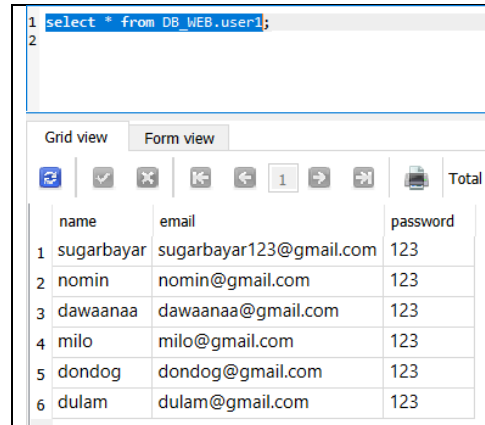
Section1		Section2																																																							
<div>Logged in :sugarbayer   <a href="#">Logout</a></div> <div>All requests for currency exchange</div> <table border="1"><thead><tr><th>Name</th><th>Date</th><th>Count</th><th>Exch</th><th>Type</th><th>Phone</th></tr></thead><tbody><tr><td>adam</td><td>20230102</td><td>200</td><td>EUR</td><td>sell</td><td>91114545</td></tr><tr><td>john</td><td>20230119</td><td>150</td><td>CNY</td><td>buy</td><td>88887777</td></tr><tr><td>sugarbayer</td><td>20230118</td><td>500</td><td>USD</td><td>buy</td><td>77567756</td></tr><tr><td>sugarbayer</td><td>20240225</td><td>100</td><td>MNT</td><td>sell</td><td>88178817</td></tr><tr><td>dondog</td><td>20231112</td><td>160</td><td>CNY</td><td>sell</td><td>85858585</td></tr></tbody></table>		Name	Date	Count	Exch	Type	Phone	adam	20230102	200	EUR	sell	91114545	john	20230119	150	CNY	buy	88887777	sugarbayer	20230118	500	USD	buy	77567756	sugarbayer	20240225	100	MNT	sell	88178817	dondog	20231112	160	CNY	sell	85858585	<div>Your requests</div> <table border="1"><thead><tr><th>Name</th><th>Date</th><th>Count</th><th>Exch</th><th>Type</th><th>Phone</th></tr></thead><tbody><tr><td>sugarbayer</td><td>20230118</td><td>500</td><td>USD</td><td>buy</td><td>77567756</td></tr><tr><td>sugarbayer</td><td>20240225</td><td>100</td><td>MNT</td><td>sell</td><td>88178817</td></tr></tbody></table>		Name	Date	Count	Exch	Type	Phone	sugarbayer	20230118	500	USD	buy	77567756	sugarbayer	20240225	100	MNT	sell	88178817
Name	Date	Count	Exch	Type	Phone																																																				
adam	20230102	200	EUR	sell	91114545																																																				
john	20230119	150	CNY	buy	88887777																																																				
sugarbayer	20230118	500	USD	buy	77567756																																																				
sugarbayer	20240225	100	MNT	sell	88178817																																																				
dondog	20231112	160	CNY	sell	85858585																																																				
Name	Date	Count	Exch	Type	Phone																																																				
sugarbayer	20230118	500	USD	buy	77567756																																																				
sugarbayer	20240225	100	MNT	sell	88178817																																																				
<div>Section3</div> <div><div>Name:</div><div>enter name</div><div>Date:</div><div>enter date</div><div>Count:</div><div>enter count</div><div>Currency:</div><div>enter exch</div><div>Type:</div><div>enter type</div><div>Phone:</div><div>enter phone</div><div>SaveDelete</div></div>																																																									

## Which?

I used SQLiteStudio (3.4.1) database in this project. And it works on my local server. Firstly, I created DB\_WEB schema. In this schema, there are two tables such as user1, and exchange1.



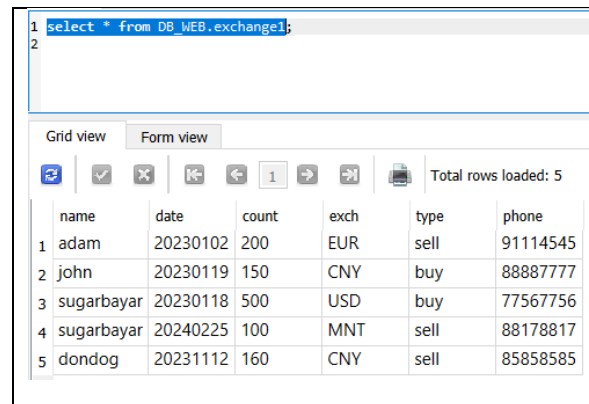
**User1** table stores information about registered users. For example, name, email, and password.



```
1 select * from DB_WEB.user1;
```

	name	email	password
1	sugarbayar	sugarbayar123@gmail.com	123
2	nomin	nomin@gmail.com	123
3	dawaanaa	dawaanaa@gmail.com	123
4	miilo	miilo@gmail.com	123
5	dondog	dondog@gmail.com	123
6	dulam	dulam@gmail.com	123

**Exchange1** table stores requests. For example, name, date, count, exch, type, and phone.



```
1 select * from DB_WEB.exchange1;
```

	name	date	count	exch	type	phone
1	adam	20230102	200	EUR	sell	91114545
2	john	20230119	150	CNY	buy	88887777
3	sugarbayar	20230118	500	USD	buy	77567756
4	sugarbayar	20240225	100	MNT	sell	88178817
5	dondog	20231112	160	CNY	sell	85858585








## How?

All materials of this project are in “login-register-app”. There are 3 files in it.

- templates folder – it includes HTML code of the website
- app.py – main python code
- DB\_WEB – it is our database file

There are 3 HTML files in the templates folder.

- login.html – HTML code of login page
- register.html – HTML code of register page
- user.html – HTML code of user page

 login-register-app	2023.01.20 14:48	File folder	
 templates	2023.01.20 14:00	File folder	
 app	2023.01.20 14:31	IPYNB File	13 KB
 DB_WEB	2023.01.20 14:08	SQLite	12 KB
 login	2023.01.04 22:11	Chrome HTML Do...	2 KB
 register	2023.01.04 23:06	Chrome HTML Do...	2 KB
 user	2023.01.20 14:00	Chrome HTML Do...	4 KB

## Description of codes?

### ➤ login HTML

All html code must be in between `<html>` and `</html>`. HTML code has two parts such as head and body.

In the head part, I wrote the character set, styles, scripts, and other information.

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Login Form</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
</head>
```

Created class, its name is a container.

```
<body>
<div class="container">
```

Created text output, text's size is h2.

```
<h2>User Login</h2>
```

The post method is to send an email, and password data to a server to check these are the in user1 table. Action name is 'login'. We will use this 'login' name in python code. Also, it will show message object. We created text output as message in python code.

```
<form action="{{ url_for('login') }}" method="post">
{% if message is defined and message %}
<div class="alert alert-warning">{{ message }}</div>
{% endif %}
```

I created two inputs. Labels are Email: and Password:. Ids are email and password. You can see placeholder texts in input section as text. Also, Login button was created. Type is submit.

Also, I created link section using href. It saved as 'register'. We will use it in python code.

```
<div class="form-group">
<label for="email">Email:</label>
<input type="email" class="form-control" id="email" name="email" placeholder="Enter email" name="email">
</div>
<div class="form-group">
<label for="pwd">Password:</label>
<input type="password" class="form-control" id="password" name="password" placeholder="Enter password" name="pswd">
</div>
<button type="submit" class="btn btn-primary">Login</button>
<p class="bottom">Don't have an account? <a class="bottom" href="{{ url_for('register') }}"> Register here</a></p>
```

### ➤ register HTML

The structure of the code is like login.html. The action name is "register", and we will use it in python code. Post is sent name, email, and password data to the database. There are 3 inputs such as name, email, and password. Each one has its own id. Also, there is a button it is called Register. Also, there is a link section, and its name is login. We will use it in python code.

```

<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Registration Form</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container">
  <h2>User Registration</h2>
  <form action="{{ url_for('register') }}" method="post">
    {% if message is defined and message %}
      <div class="alert alert-warning">{{ message }}</div>
    {% endif %}
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" class="form-control" id="name" name="name" placeholder="Enter name" name="name">
    </div>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" name="email" placeholder="Enter email" name="email">
    </div>
    <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="password" name="password" placeholder="Enter password" name="pwd">
    </div>
    <button type="submit" class="btn btn-primary">Register</button>
    <p class="bottom">Already have an account? <a class="bottom" href="{{url_for('login')}}"> Login here</a></p>
  </form>
</div>
</body>
</html>

```

### ➤ User HTML

The head of user.html is like the previous one. There is a style of table, and it is created in between <style> and </style>. There are fonts, padding size, font weight, and many color codes.

```

<style>
table {
  border-collapse: collapse;
  font-family: Tahoma, Geneva, sans-serif;
}
table td {
  padding: 11px;
}
table thead td {
  background-color: #54585d;
  color: #ffffff;
  font-weight: bold;
  font-size: 11px;
  border: 1px solid #54585d;
}
table tbody td {
  color: #636363;
  font-size: 11px;
  border: 1px solid #dddfel;
}
table tbody tr {
  background-color: #ffffff;
}
table tbody tr:nth-child(odd) {
  background-color: #ffffff;
}
</style>

```

You will see the user profile text output and its size is h1. Also, you will see “Logged in:” text. And, after it, you will see session.name. We will use it in python code. In other words, it will be shown who is logged in. After it, you will see logout link and its name is ‘logout’. We will use it in python code. Also, you will see text messages and their name is message. We will use it in python code. There is h3-sized text.

```

<div class="container">
  <div class="row">
    <h1>User Profile</h1>
  </div>
  <br>
  <div class="row">
    Logged in : <strong>{{session.name}} | <a href="{{ url_for('logout') }}"> Logout</a>
  </div>
  <br>
  {% if message is defined and message %}
    <div class="alert alert-warning">{{ message }}</div>
  {% endif %}
  <br>
  <div class="row">
    <h3>All requests for currency exchange</h3>
  </div>
</div>

```

This section has 2 tables. So, I must give them different id names. This table's id is "table1". In between <thead> and </thead>, I wrote the columns name of table1. And after it, I wrote code to create rows of table 1. It is a loop and it will be shown row in each row. We will use "rows" in python code.

```
<table border="1" id='table1'>
  <thead>
    <td>Name</td>
    <td>Date</td>
    <td>Count</td>
    <td>Exch</td>
    <td>Type</td>
    <td>Phone</td>
  </thead>
  {% for row in rows %}
    <tr>
      <td>{{row['name']}}</td>
      <td>{{row['date']}}</td>
      <td>{{row['count']}}</td>
      <td>{{row['exch']}}</td>
      <td>{{row['type']}}</td>
      <td>{{row['phone']}}</td>
    </tr>
  {% endfor %}
</table>
```

This action's name is "update" and we will use it in python code. Also, this section sent the below data to the database. There are 6 inputs. And there are two buttons. These two buttons have different actions. Therefore, their names are different such as save\_ex and send\_ex.

```
<div class="row">
  <form action="{{ url_for('update') }}" method="post">

    <div class="form-group">
      <label for="name1">Name:</label>
      <input type="text" class="form-control" id="name1" name="name1" placeholder="enter name" name="name1">
    </div>

    <div class="form-group">
      <label for="date1">Date:</label>
      <input type="text" class="form-control" id="date1" name="date1" placeholder="enter date" name="date1">
    </div>

    <div class="form-group">
      <label for="count1">Count:</label>
      <input type="text" class="form-control" id="count1" name="count1" placeholder="enter count" name="count1">
    </div>

    <div class="form-group">
      <label for="exch1">Currency:</label>
      <input type="text" class="form-control" id="exch1" name="exch1" placeholder="enter exch" name="exch1">
    </div>

    <div class="form-group">
      <label for="type1">Type:</label>
      <input type="text" class="form-control" id="type1" name="type1" placeholder="enter type" name="type1">
    </div>

    <div class="form-group">
      <label for="phone1">Phone:</label>
      <input type="text" class="form-control" id="phone1" name="phone1" placeholder="enter phone" name="phone1">
    </div>

    <button type="submit" class="btn btn-primary" name="save_ex">Save</button>
    <button type="submit" class="btn btn-primary" name="send_ex">Delete</button>
  </form>
</div>
```

There is h2-sized text. Also, there is a table, its id is table2. We will use this id in python code. It will be shown a row in each entry. We will use entries in python code.

```
<h2>Your requests</h2>
<div class="row">
  <table border="1" id='table2'>
    <thead>
      <td>Name</td>
      <td>Date</td>
      <td>Counts</td>
      <td>Exch</td>
      <td>Type</td>
      <td>Phone</td>
    </thead>
    {% for row in entries %}
      <tr>
        <td>{{row['name']}}</td>
        <td>{{row['date']}}</td>
        <td>{{row['count']}}</td>
        <td>{{row['exch']}}</td>
        <td>{{row['type']}}</td>
        <td>{{row['phone']}}</td>
      </tr>
    {% endfor %}
  </table>
</div>
```

➤ app python

First, I imported libraries to use. I mentioned the directory where HTML codes are. After it, I connected python to the database, and it is saved as con.

```

from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysql import MySQL
import MySQLdb.cursors
import re
import sqlite3

app = Flask(__name__, template_folder='C:/Users/User/Desktop/login-register-app/templates')
app.secret_key = 'xyzsdfg'

def get_db_connection():
    con = sqlite3.connect('C:/Users/User/Desktop/login-register-app/DB_WEB.db')
    con.row_factory = sqlite3.Row
    return con

```

Login() - In html, I wrote action="{ { url\_for('login') } }". This section will describe what to do in login. First, I created a login function and message object. Message is an empty string object, and it will be shown on the user and login pages. Email is the input value of email. And the password is the input value of the password. Con is a connection with the database. In cursor.execute, I select rows from user1 table whose email is the same as the input and whose password is the same as the input. It is saved as a user object. If the session's name and email are the same as the user's name and email, create rows which are select all rows from exchange1. Also, it will create rows1 table which is data from exchange1 table whose name is the same as the user's name. After it, you will be user.html and rows are rows, rows1 is entries, message is message. In other words, rows and rows1 is the value of table1 and table2. The

If the user enters wrong email or password, you will see login.html and you will receive 'Please, enter correct email/password!' message.

Update() - in this section, we will insert a new request to the exchange1 table, or delete an old request from exchange1 table. Each input has its own object names such as in\_name, and in\_date. If the user clicks on the save button, it means 'save\_ex' object is in request.form. If 'save\_ex' is in request.form, insert inputs to exchange1 table. If the user clicks on the delete button, delete a row from exchange1 that is the same as the inputs. You will see login.html.

```

@app.route('/')
@app.route('/login', methods=['POST', 'GET'])
def login():
    message = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        #
        in2_name = request.form['name2']
        email = request.form['email']
        password = request.form['password']
        con = get_db_connection()
        cursor = con.cursor()
        cursor.execute('SELECT * FROM user1 WHERE email = ? AND password = ?', (email, password,))
        con.commit()
        user = cursor.fetchone()
        if user:
            session['loggedin'] = True
            session['name'] = user['name']
            session['email'] = user['email']
            message = 'Logged in successfully !'
            name_file = user['name']
            rows = cursor.execute('select * from exchange1').fetchall()
            rows1 = cursor.execute('select * from exchange1 where name=?', (name_file,)).fetchall()
            return render_template('user.html', rows=rows, entries=rows1, message=message)
        else:
            message = 'Please enter correct email / password !'
    return render_template('login.html', message=message)

@app.route('/', methods=['POST'])
def update():
    message = ''
    if request.method == 'POST' and 'name1' in request.form and 'date1' in request.form and 'count1' in request.form
    and 'exch1' in request.form and 'type1' in request.form and 'phone1' in request.form:
        in_name = request.form['name1']
        in_date = request.form['date1']
        in_count = request.form['count1']
        in_exch = request.form['exch1']
        in_type = request.form['type1']
        in_phone = request.form['phone1']
        con = get_db_connection()
        cursor = con.cursor()
        if 'save_ex' in request.form:
            cursor.execute("INSERT INTO exchange1 ('name','date','count','exch','type','phone') VALUES (?, ?, ?, ?, ?, ?)",
                (in_name, in_date, in_count, in_exch, in_type, in_phone,))
        elif 'send_ex' in request.form:
            cursor.execute("DELETE from exchange1 where name = ? and date=? and count=? and exch=? and type=? and phone=?",
                (in_name, in_date, in_count, in_exch, in_type, in_phone,))
        con.commit()
        message = ''
    elif request.method == 'POST':
        message = ''
    return render_template('login.html', message=message)

```

Logout() – If you click on logout, you will see login.html. the loggedin and email are removed from the object.

Register() – If you don't fill in all 3 inputs such as name, password, and email, you will see "Please fill out the form!" message. If you didn't use @ in email section, you will see "invalid email address" message. If you use the email which you used before, you will see "account already exists !" message.

After filling in all 3 inputs, insert these data into the user1 table. And you will see "you have successfully registered !".

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

@app.route('/register', methods=['POST', 'GET'])
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']
        con = get_db_connection()
        cursor = con.cursor()
        cursor.execute('SELECT * FROM user1 WHERE email = ?', (email,))
        con.commit()
        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 user1 ('name','email','password') VALUES ( ?, ?, ?)", (userName, email, password,))
            con.commit()
            message = 'You have successfully registered !'
    elif request.method == 'POST':
        message = 'Please fill out the form !'
    return render_template('register.html', message = message)

if __name__ == "__main__":
    app.run()
```

Sugarbayar Enkhbayar

456296

s.enkhbayar@student.uw.edu.pl