# Task Title: Signup Form with SQLite Integration

**Full Name**: Jacob Sam Jose

**Batch:** June 2025

**Internship ID:** VERVEOX25PY35

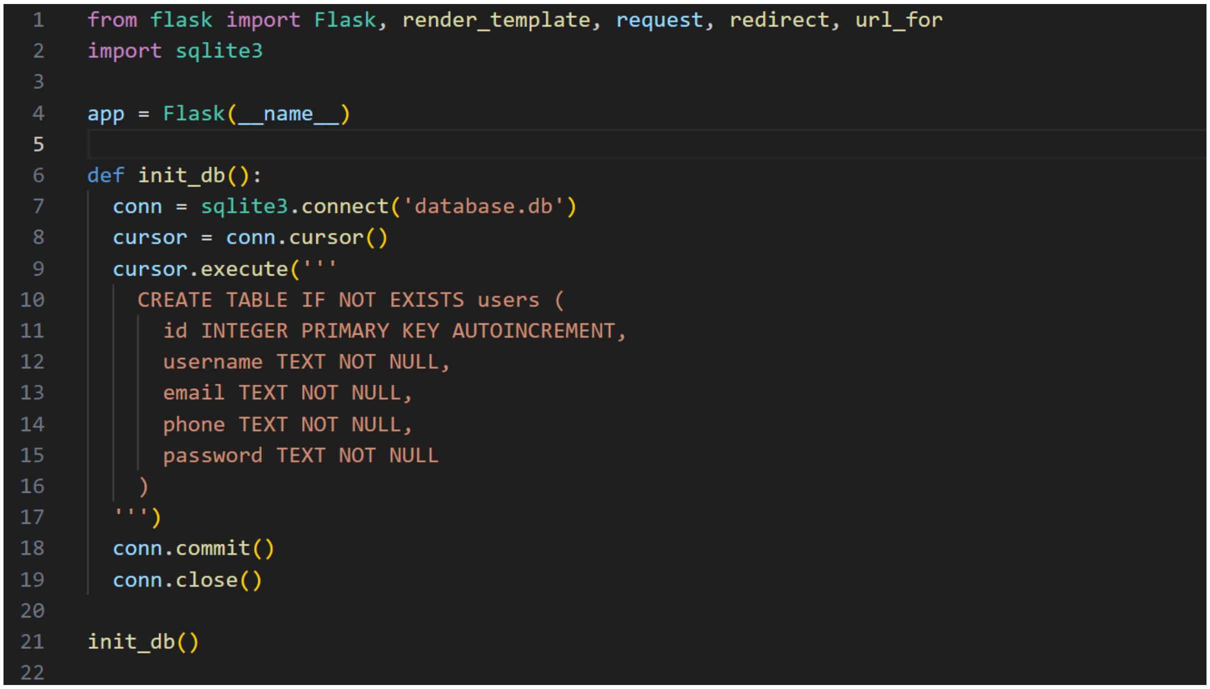
**Submission Date:** 26/ 06/2025

x

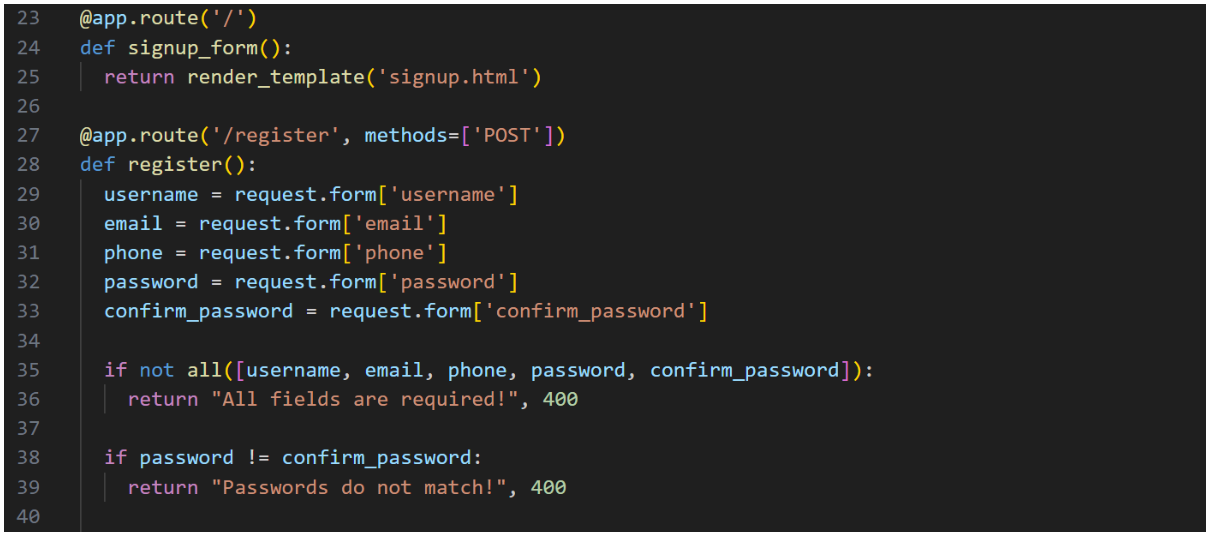
## 1. Objective

This signup form is designed to securely collect user information such as username, email, phone number, and password. It uses Flask to handle form submissions, validates the input, and stores the data in an SQLite database for future use.

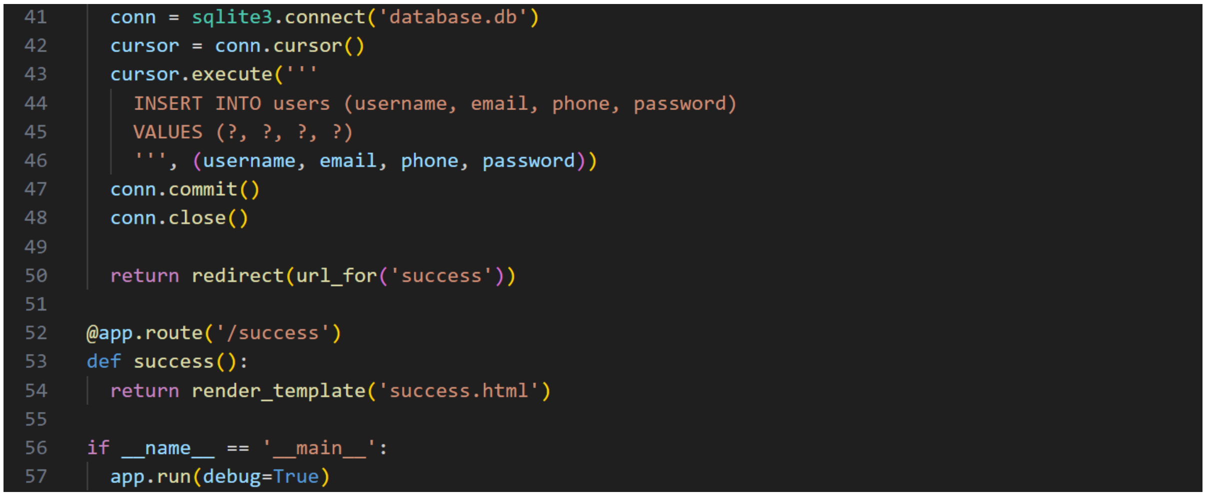
## 2. Source Code



```python
from flask import Flask, render_template, request, redirect, url_for
import sqlite3

app = Flask(__name__)

def init_db():
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT NOT NULL,
            email TEXT NOT NULL,
            phone TEXT NOT NULL,
            password TEXT NOT NULL
        )
    ''')
    conn.commit()
    conn.close()

init_db()

```

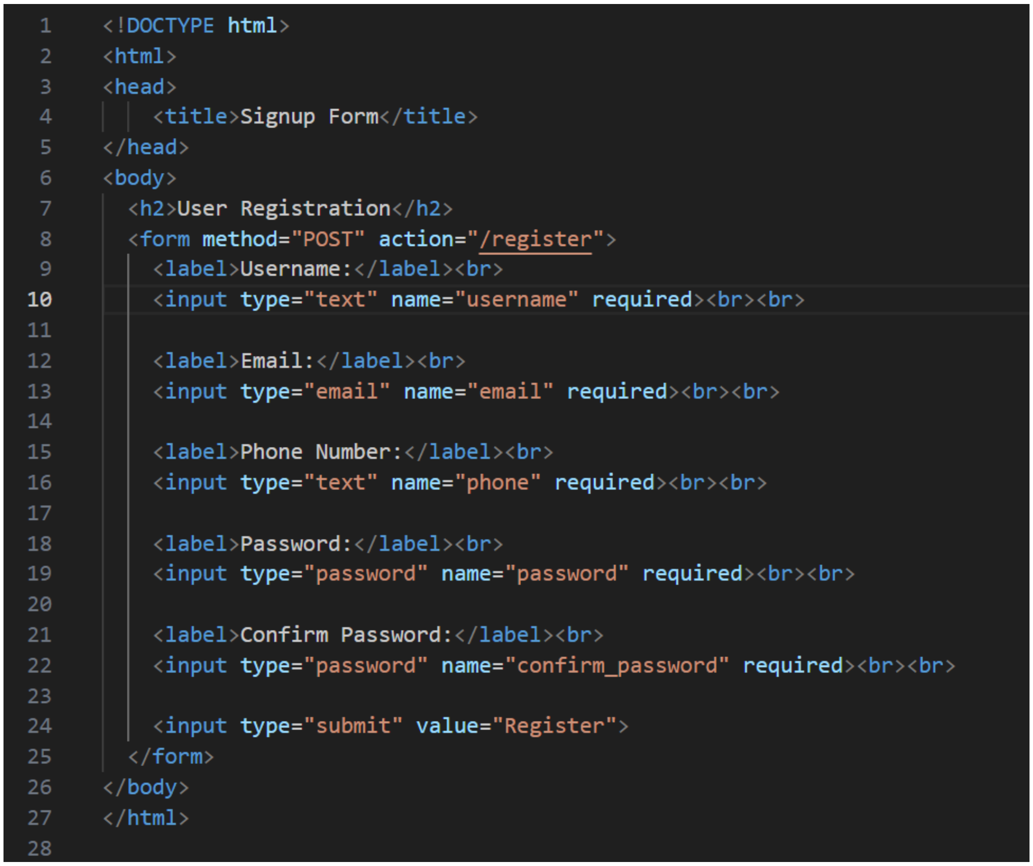


```python
@app.route('/')
def signup_form():
    return render_template('signup.html')

@app.route('/register', methods=['POST'])
def register():
    username = request.form['username']
    email = request.form['email']
    phone = request.form['phone']
    password = request.form['password']
    confirm_password = request.form['confirm_password']

    if not all([username, email, phone, password, confirm_password]):
        return "All fields are required!", 400

    if password != confirm_password:
        return "Passwords do not match!", 400

```



```
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO users (username, email, phone, password)
        VALUES (?, ?, ?, ?)
        ''', (username, email, phone, password))
    conn.commit()
    conn.close()

    return redirect(url_for('success'))

@app.route('/success')
def success():
    return render_template('success.html')

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

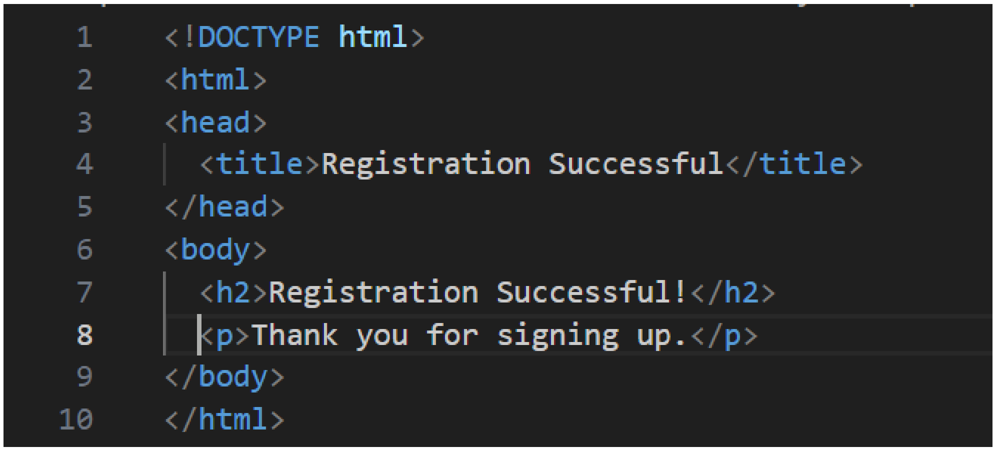
## HTML FORM



```
<!DOCTYPE html>
<html>
<head>
    <title>Signup Form</title>
</head>
<body>
  <h2>User Registration</h2>
  <form method="POST" action="/register">
    <label>Username:</label><br>
    <input type="text" name="username" required><br><br>

    <label>Email:</label><br>
    <input type="email" name="email" required><br><br>

    <label>Phone Number:</label><br>
    <input type="text" name="phone" required><br><br>

    <label>Password:</label><br>
    <input type="password" name="password" required><br><br>

    <label>Confirm Password:</label><br>
    <input type="password" name="confirm_password" required><br><br>

    <input type="submit" value="Register">
  </form>
</body>
</html>
```

x

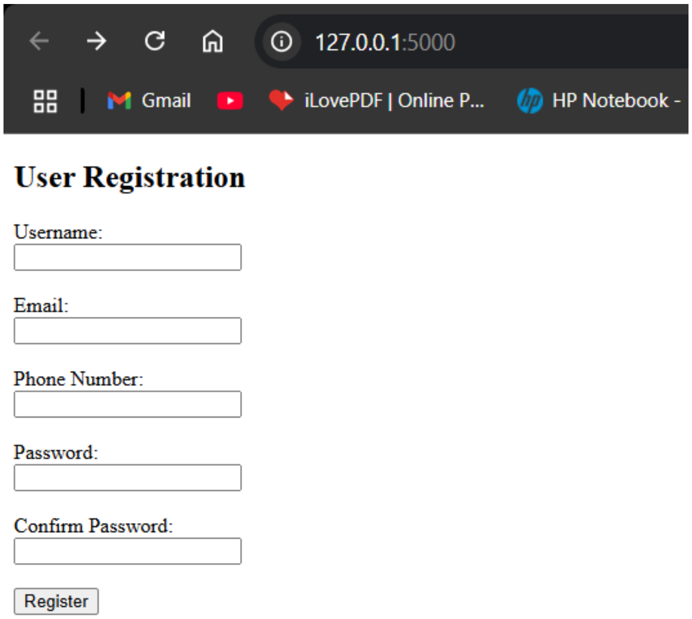
## Success Page



```
<!DOCTYPE html>
<html>
<head>
  <title>Registration Successful</title>
</head>
<body>
  <h2>Registration Successful!</h2>
  <p>Thank you for signing up.</p>
</body>
</html>
```

# Program Output

### 1. SignUp Form



### User Registration

Username:

Email:

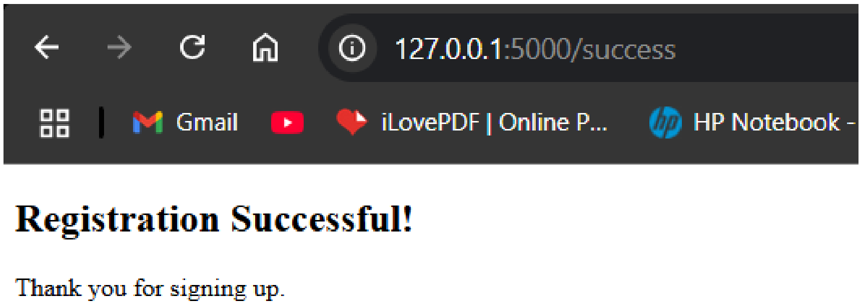
Phone Number:

Password:

Confirm Password:

x

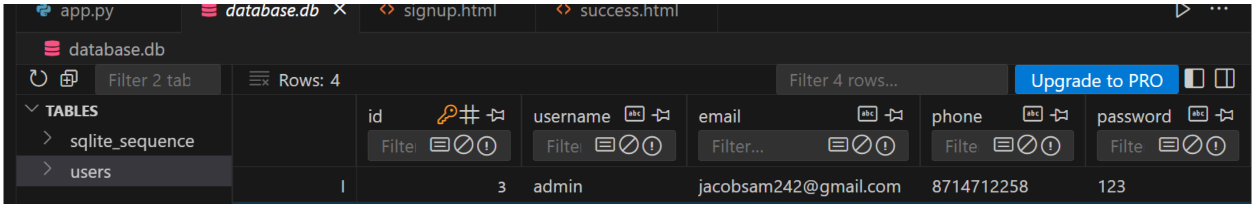
## 2. Success message page



# Registration Successful!

Thank you for signing up.

## 2. SQLite Database



Rows: 4

|   | id | username | email | phone | password |
|---|---|---|---|---|---|
| I | 3 | admin | jacobsam242@gmail.com | 8714712258 | 123 |

x

## 4 .Code Explanation

The Flask route / displays the signup form to the user, while /success shows a confirmation message after successful registration.

When the form is submitted, Flask validates that all fields are filled and checks if the passwords match. If valid, the user data is inserted into the SQLite database.

After storing the data, Flask uses redirect() to navigate the user to the /success page.

## 5. Challenges Faced

**Form not submitting properly**: This was due to missing the method="POST" attribute in the HTML form. After adding it the form data was sent to the backend correctly.

**Database errors**: At first, the users table didn't exist, which caused insertion errors. This was fixed by creating the table using a query with CREATE TABLE IF NOT EXISTS.

**Redirect not working**: Initially, the redirect to /success failed because the route name was incorrect. Using redirect(url_for('success')) resolved the issue.