TASK 1:

Prepared by : KISHORE S

Title: Flask-Based Signup and Login System Report

Objective:

To design and implement a simple user authentication system using Python Flask that supports user **signup**, **login**, and a **welcome page** with GIF animation.

Technologies Used:

- Python 3
- Flask
- HTML/CSS
- JavaScript (for loading effects)

Folder Structure:

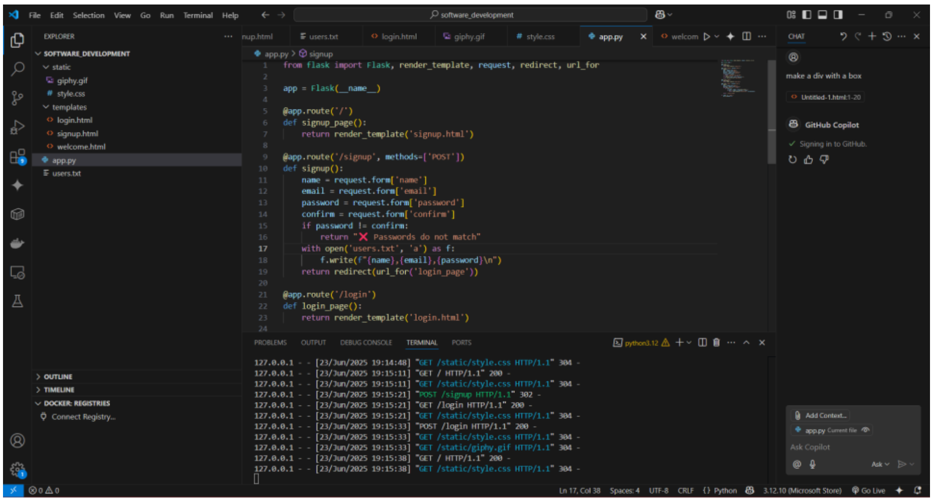
```
software_development/
 ├ static/
|   ├ giphy.gif
|  └ style.css
 ├ templates/
|   ├ login.html
|   ├ signup.html
|  └ welcome.html
 ├ app.py
└ users.txt
```

Description:

1. **Signup Page:**
   - Allows users to register with name, email, password, and confirm password.
   - Stores user data in users.txt.
   - Validates matching passwords.
2. **Login Page:**
   - Validates credentials by reading from users.txt.
   - If successful, redirects to a welcome page.
3. **Welcome Page:**
   - Greets user by email.
   - Displays an animated GIF (stored in static/giphy.gif).

1. **Code View:** *(Place after the "Folder Structure" section)*

   app.py file in VS Code showing routing and logic.

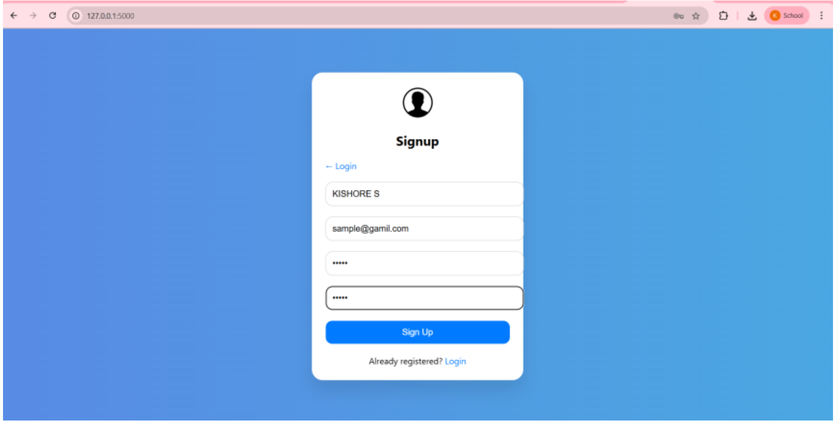


```python
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

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

@app.route('/signup', methods=['POST'])
def signup():
    name = request.form['name']
    email = request.form['email']
    password = request.form['password']
    confirm = request.form['confirm']
    if password != confirm:
        return "❌ Passwords do not match"
    with open('users.txt', 'a') as f:
        f.write(f"{name},{email},{password}\n")
    return redirect(url_for('login_page'))

@app.route('/login')
def login_page():
    return render_template('login.html')
```

```
127.0.0.1 - - [23/Jun/2025 19:14:48] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 19:15:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jun/2025 19:15:11] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 19:15:21] "POST /signup HTTP/1.1" 302 -
127.0.0.1 - - [23/Jun/2025 19:15:21] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [23/Jun/2025 19:15:21] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 19:15:33] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [23/Jun/2025 19:15:33] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 19:15:33] "GET /static/giphy.gif HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 19:15:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jun/2025 19:15:38] "GET /static/style.css HTTP/1.1" 304 -
```

2. **Signup Page:** *(Place after the "Signup Page" description)*

   Signup form in browser.



127.0.0.1:5000

Signup

KISHORE S

sample@gamil.com

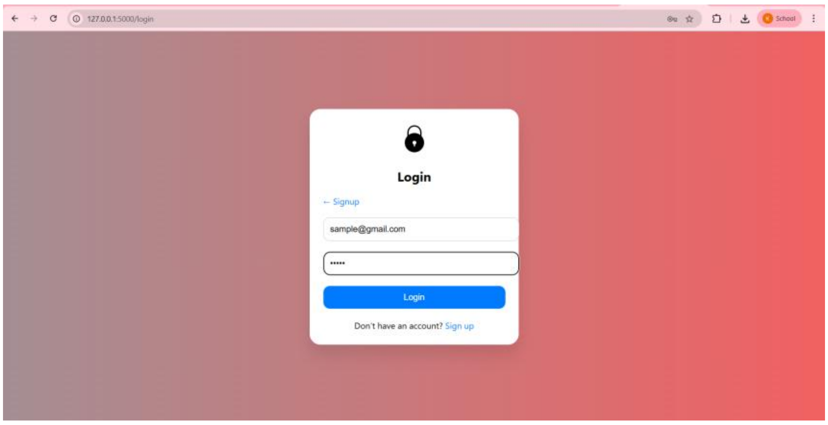
•••••

•••••

Already registered? Login

3. **Login Page:** *(Place after the "Login Page" description)*

   Login form in browser.



127.0.0.1:5000/login

Login

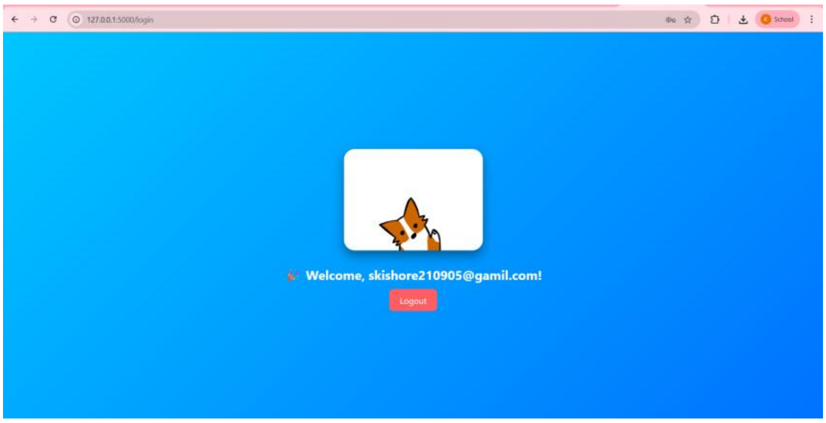
sample@gmail.com

•••••

Don't have an account? Sign up

4. **Welcome Page:** *(Place after the "Welcome Page" description)*

   Welcome message with working giphy.gif animation.



127.0.0.1:5000/login

🎉 Welcome, skishore210905@gamil.com!

Conclusion:

This project demonstrates a minimal full-stack web app using Flask. It helps understand basic routing, form handling, and static asset management in web development.