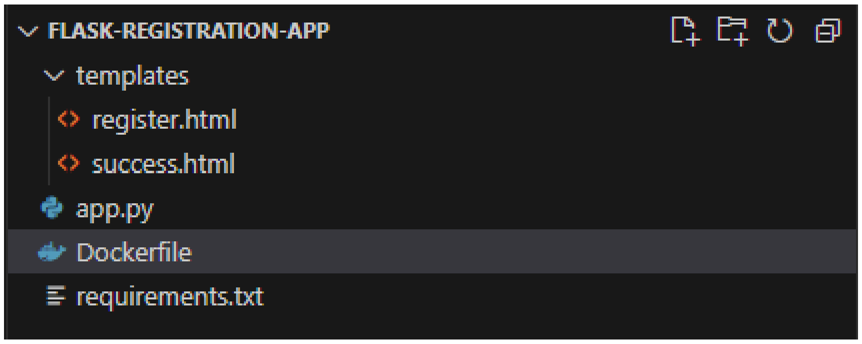
Jawahar Education Society's
A.C. Patil College of Engineering, Kharghar
Department of CSE (IoT-CS BC)

Experiment No. 1

Aim: Write code for a simple user registration form for an event.

Description: Flask registration app that demonstrates user registration with a form and basic validation. This app will use Flask to render a registration form, handle form submission, and display a success message.

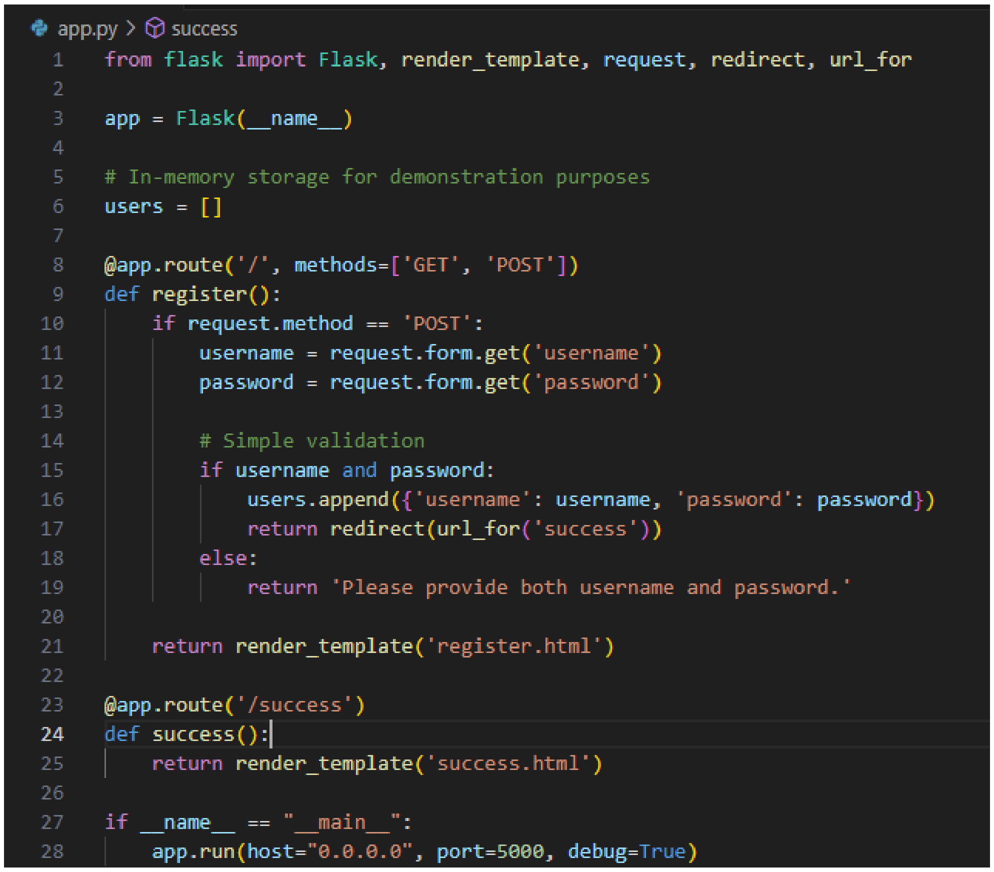
Directory Structure



```
FLASK-REGISTRATION-APP
  templates
    register.html
    success.html
  app.py
  Dockerfile
  requirements.txt
```

1. Create app.py

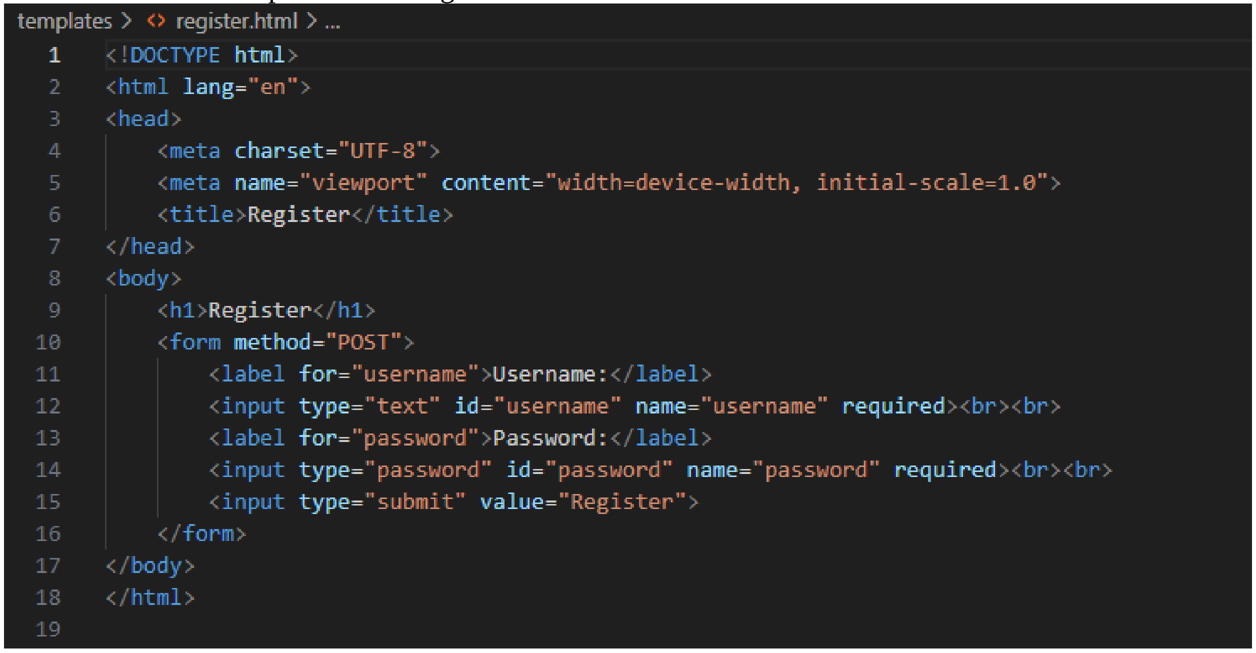
This is the main Python file for your Flask application:



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

app = Flask(__name__)

# In-memory storage for demonstration purposes
users = []

@app.route('/', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        # Simple validation
        if username and password:
            users.append({'username': username, 'password': password})
            return redirect(url_for('success'))
        else:
            return 'Please provide both username and password.'

    return render_template('register.html')

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

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

2. Create templates/register.html

This is the HTML template for the registration form:

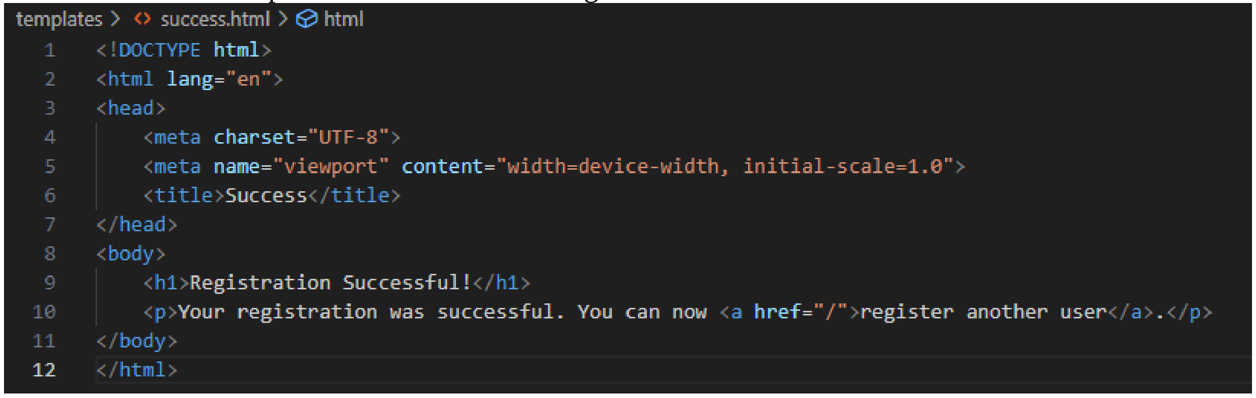


templates > register.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
</head>
<body>
    <h1>Register</h1>
    <form method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Register">
    </form>
</body>
</html>

```

3. Create templates/success.html

This is the HTML template for the success message:

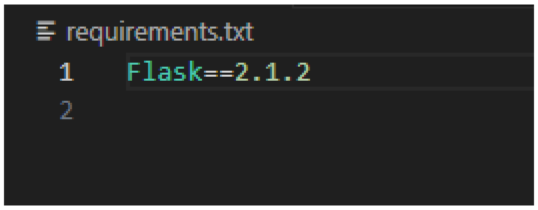


templates > success.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Success</title>
</head>
<body>
    <h1>Registration Successful!</h1>
    <p>Your registration was successful. You can now <a href="/">register another user</a>.</p>
</body>
</html>
```

4. Create requirements.txt

List the dependencies required by the app:

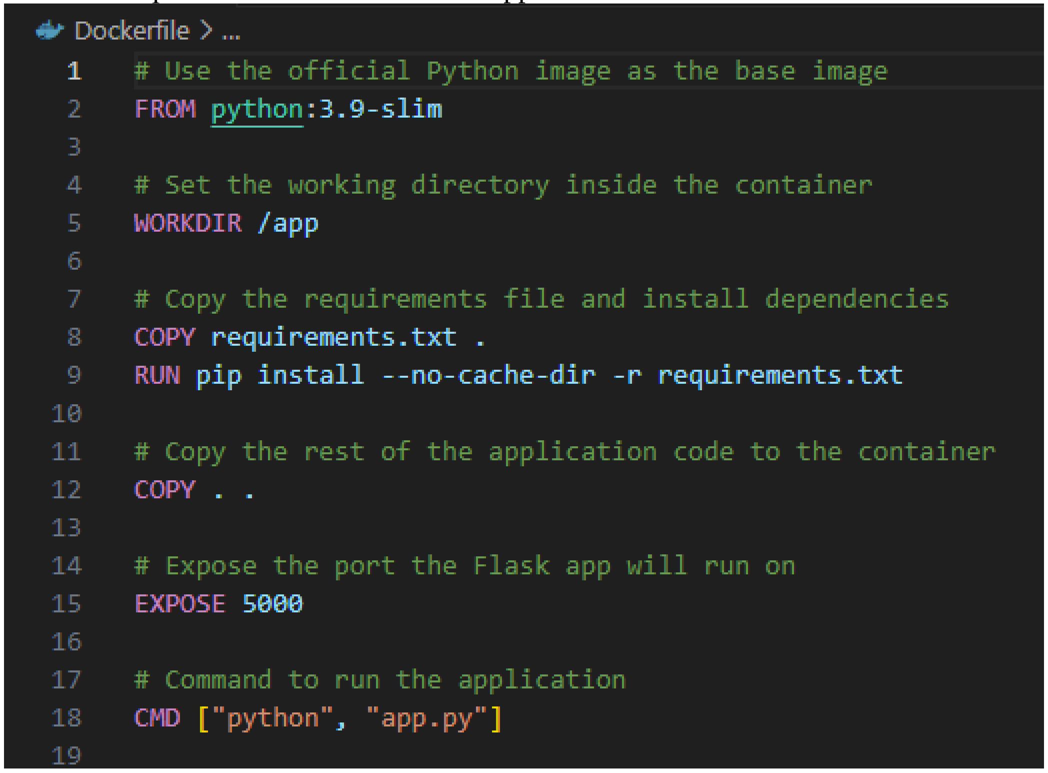


requirements.txt

```
Flask==2.1.2

```

5. Dockerize the Application

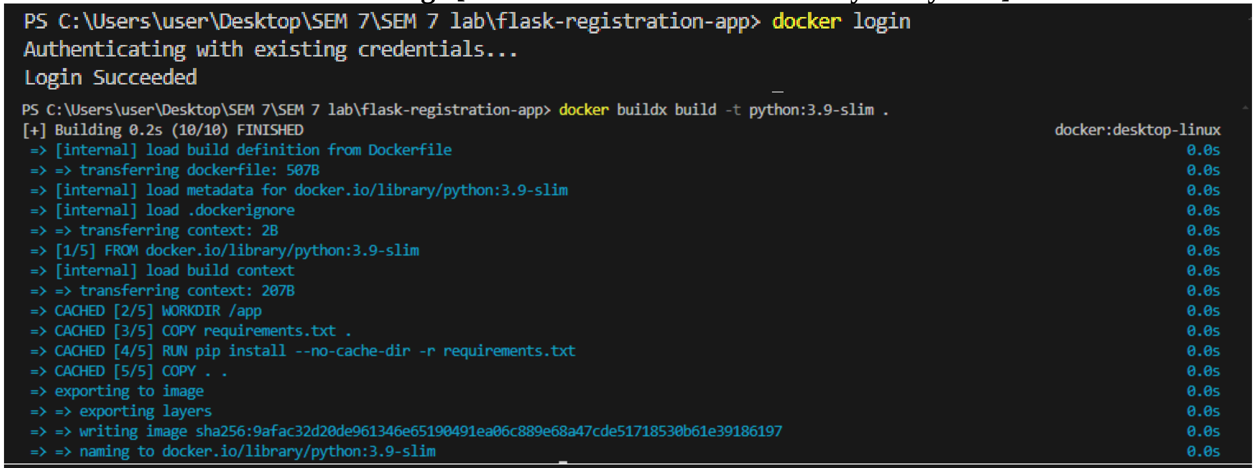
Here’s a simple Dockerfile for this Flask application:



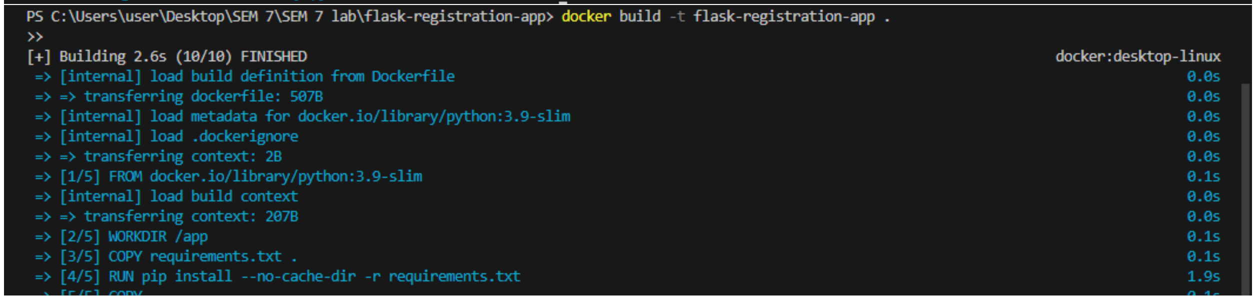
Dockerfile > ...

```dockerfile
# Use the official Python image as the base image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code to the container
COPY . .

# Expose the port the Flask app will run on
EXPOSE 5000

# Command to run the application
CMD ["python", "app.py"]

```

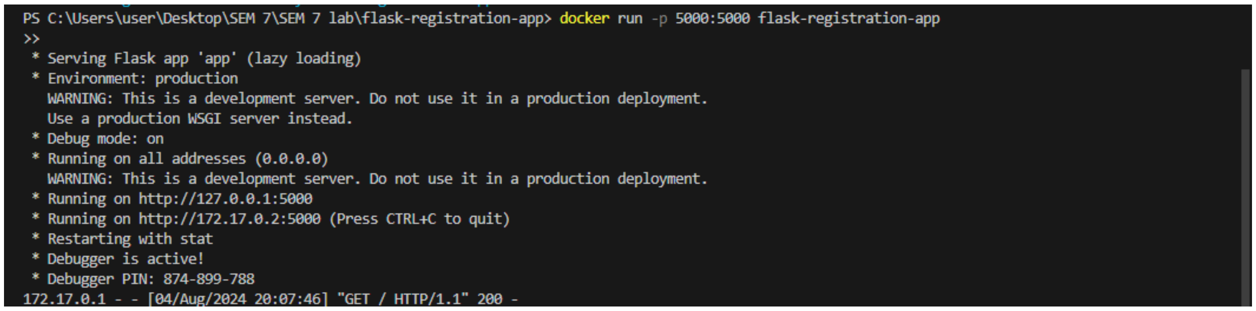
6. Build and Run the Docker Image [Docker should be installed on your system]



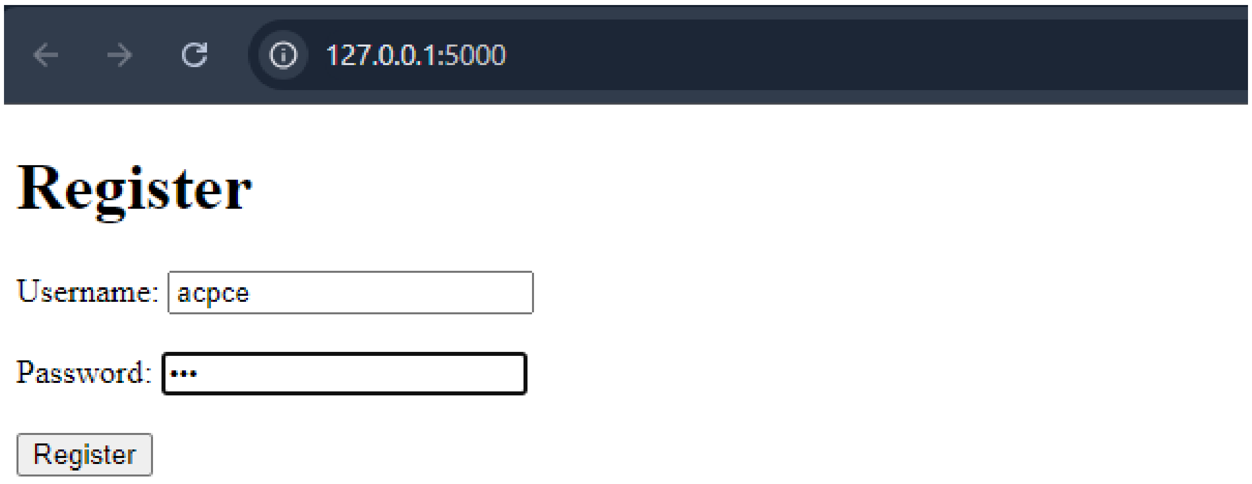
```
PS C:\Users\user\Desktop\SEM 7\SEM 7 lab\flask-registration-app> docker login
Authenticating with existing credentials...
Login Succeeded

PS C:\Users\user\Desktop\SEM 7\SEM 7 lab\flask-registration-app> docker buildx build -t python:3.9-slim .
[+] Building 0.2s (10/10) FINISHED                                                         docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                       0.0s
 => => transferring dockerfile: 507B                                                                       0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                         0.0s
 => [internal] load .dockerignore                                                                          0.0s
 => => transferring context: 2B                                                                            0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim                                                           0.0s
 => [internal] load build context                                                                          0.0s
 => => transferring context: 207B                                                                          0.0s
 => CACHED [2/5] WORKDIR /app                                                                              0.0s
 => CACHED [3/5] COPY requirements.txt .                                                                   0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                                        0.0s
 => CACHED [5/5] COPY . .                                                                                  0.0s
 => exporting to image                                                                                     0.0s
 => => exporting layers                                                                                    0.0s
 => => writing image sha256:9afac32d20de961346e65190491ea06c889e68a47cde51718530b61e39186197               0.0s
 => => naming to docker.io/library/python:3.9-slim                                                         0.0s
```



```
PS C:\Users\user\Desktop\SEM 7\SEM 7 lab\flask-registration-app> docker build -t flask-registration-app .
>>
[+] Building 2.6s (10/10) FINISHED                                                         docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                       0.0s
 => => transferring dockerfile: 507B                                                                       0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                         0.0s
 => [internal] load .dockerignore                                                                          0.0s
 => => transferring context: 2B                                                                            0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim                                                           0.1s
 => [internal] load build context                                                                          0.0s
 => => transferring context: 207B                                                                          0.0s
 => [2/5] WORKDIR /app                                                                                     0.1s
 => [3/5] COPY requirements.txt .                                                                          0.1s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                                               1.9s
```



```
PS C:\Users\user\Desktop\SEM 7\SEM 7 lab\flask-registration-app> docker run -p 5000:5000 flask-registration-app
>>
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 874-899-788
172.17.0.1 - - [04/Aug/2024 20:07:46] "GET / HTTP/1.1" 200 -
```

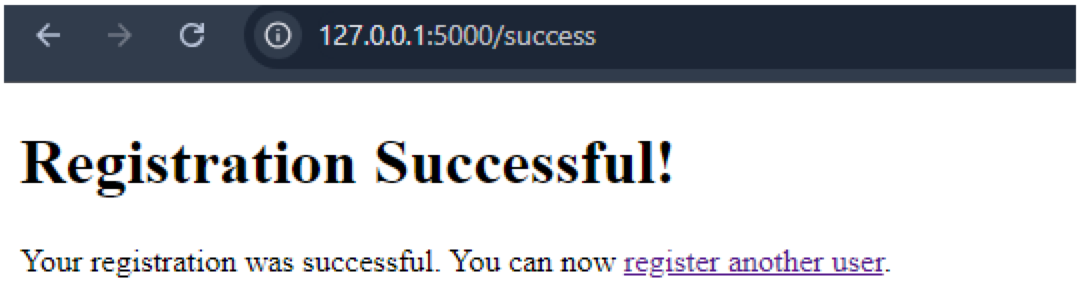


127.0.0.1:5000

Register

Username: acpce

Password: •••



127.0.0.1:5000/success

Registration Successful!

Your registration was successful. You can now register another user.

Accessing the Application

Navigate to http://localhost:5000 in your web browser to see the registration form. You can register a user, and upon success, you will be redirected to a success page.

This example provides a basic registration form and uses in-memory storage. For a real application, you would typically store user data in a database and implement additional security measures.