

2_1. 일반 로그인 화면, 프로필 화면 창 구현.

2.1. OAUTH 회원가입_ 로그인 화면 창 만들기.

2.1.1. 간단한 회원가입 기능 구현하기. **with** flask 서버, mysql, html 코드.

회원가입 창을 만들 때 쓴 html 코드.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>회원가입</title>
</head>
<body>
  <h2>회원가입</h2>
  <form action="/register" method="POST">

    <label for="user_id">아이디:</label>
    <input type="text" name="user_id" required><br><br>

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

    <label for="nickname">닉네임:</label>
    <input type="text" name="nickname" required><br><br>

    <button type="submit">회원가입</button>
  </form>
</body>
</html>
```

위 html 코드에서 배울 점.

<1_ required_

```
<input type="text" name="user_id" required>
```

- **역할:** 사용자가 해당 입력 필드를 **비워둔 채로 제출, submit하지 못하게** 브라우저가 경고 메시지를 띄우고 제출을 막습니다.
- HTML5에서 제공합니다.

<2_method="POST"

```
<form action="/register" method="POST">
```

- **역할:** 폼 데이터를 전송할 때 사용하는 **HTTP 요청 방식**을 지정합니다.
- **POST** 는 데이터를 *본문(body)*에 담아서 서버로 전송합니다.
- 보통 **회원가입, 로그인, 글 작성** 등 데이터를 서버에 **변경/저장**할 때 사용합니다.

<3_ action="/register"

```
<form action="/register" method="POST">
```

- **역할:** 폼 데이터를 전송할 *서버 주소(경로)*를 지정합니다.
- 여기서는 사용자가 입력한 회원가입 정보를 `/register` 라는 URL로 전송합니다.
- Flask에서는 이 경로에 해당하는 **라우트 핸들러 함수**를 만들어야 합니다.
- → 즉, Flask에서 `/register` 라우트에서 한 `POST` 요청을 다음과 같은 파이썬 코드로써 받습니다.

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

app = Flask(__name__)

@app.route('/register', methods=['POST'])
def register():
    # 여기서 폼 데이터를 처리합니다.
    # HTML 폼에서 전달된 데이터 가져오기
    user_id = request.form['user_id']
    password = request.form['password']
    nickname = request.form['nickname']

    # 여기에 DB 저장, 중복 체크, 비밀번호 암호화 등을 구현 가능

    print(f"아이디: {user_id}, 비밀번호: {password}, 닉네임: {nickname}")

    # 예: 회원가입 후 로그인 페이지로 이동
    return redirect('/login')
```

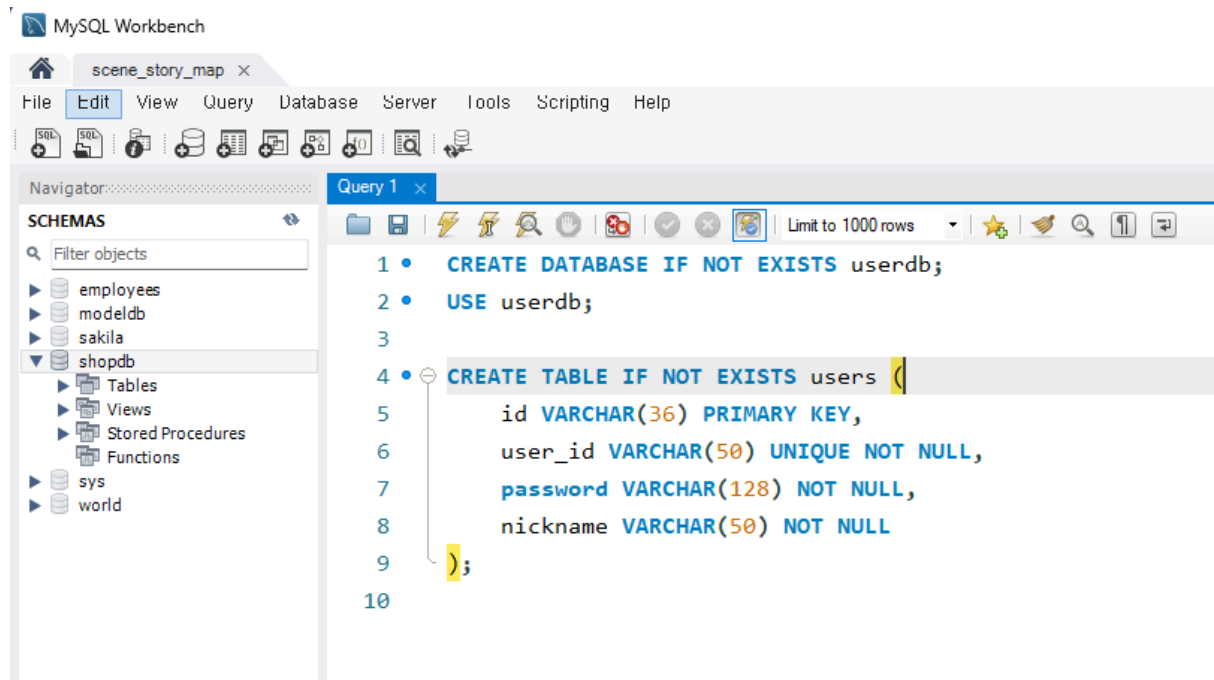
<_ request.form['user_id'] :
 폼에서 전송된 값 읽기 (
`name="user_id"` 인 input에서 가져옴)

<_ redirect('/login') : 읽기 및 읽은 데이터 저장 및 출력 후 로그인 페이지로 이동 (필요 시
 변경 가능)

2.1.1.1: 실행 예시 구현하기_
 웹사이트에서

ID, 비밀번호, 닉네임을 입력하고 회원가입 버튼을 누르면 Flask 서버가 이 데이터를 받아 MySQL 데이터베이스에 **사용자 객체**를 추가_를 구현하기.

<1> 다음의 mysql 코드를 실행해 userdb 라는 이름의 데이터 베이스를 생성하기.

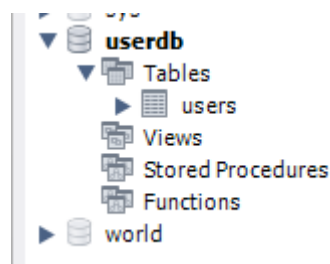


위 코드가 하는 일_ userdb 란 이름의 데이터 베이스 생성 후, 해당 데이터 베이스에 유저의 고유 id 값을 필수 키 속성으로 삼는 users 객체 테이블을 추가했습니다.

이 테이블 내부에는 VARCHAR 타입의 임의 텍스트를 50 자까지 저장해 주며 다른 객체의 속성 값과 중복될 수 없는 unique 한 user_id 값과 128 자까지 지정 가능한 비밀번호, 50자 까지 지정가능한 닉네임이 NOT NULL: NULL 불가 상태로써 속성 지정되고 있습니다.

그러나, 여러분은 위의 사진대로 작성하되, 모든 VARCHAR 데이터 타입을 갖는 속성의 최대 저장 글자 수를 전부 500 자로 통일해서 설정하시기 바랍니다. 충분한 길이를 허용하지 않으면 나중에 회원가입을 할 때 오류를 빈번히 만들 수 있습니다 😊

실행 결과_ userdb 가 users 테이블을 가진 채로 생성되는 것에 성공했습니다!



<2> 다음의 python 코드를 `app.py` 란 파일명으로 지정해서 저장한 후,

`"your_mysql_user"` 와 `"your_mysql_password"` 부분은 자신의 MySQL 계정의 user connection 이름과 비밀번호에 맞게 수정합니다. 그리고 코드를 실행, 디버깅합니다.

```
from flask import Flask, request, render_template, redirect
import mysql.connector
import uuid
from werkzeug.security import generate_password_hash

app = Flask(__name__) # ← 로컬서버에서 실행할 웹 flask 서버 프로그램 객체를 생성
# Flask 클래스의 인스턴스를 생성하여 app이라는 이름으로 저장합니다.
# app 객체는 Flask 웹 애플리케이션 전체를 대표하는 객체입니다.
# __name__은 현재 모듈의 이름을 나타내며, Flask가 현재 파일을 애플리케이션의 시작점

# MySQL 연결 설정
db = mysql.connector.connect(
    host="127.0.0.1", # ← 당신의 MySQL DB 의 host 주소 값(connection 정보에서 host)
    user="root",      # ← 당신의 MySQL 사용자명
    password="*****", # ← 당신의 MySQL 비밀번호
    database="userdb" # ← 정보를 저장할 MySQL DB 의 이름
)

# 일반 커서를 생성 (딕셔너리 형태로 결과를 받을 필요 없음)
cursor = db.cursor()

# 루트 URL ('/')로 접속 시 실행될 뷰 함수 등록
@app.route('/')
def index():
    # 'register.html' 템플릿을 클라이언트에게 렌더링하여 반환
    return render_template('register.html')

# '/register' 경로로 POST 요청이 들어왔을 때 실행되는 회원가입 처리 함수
@app.route('/register', methods=['POST'])
def register():
    # HTML 폼에서 전송된 사용자 입력 데이터 추출
```

```

user_id = request.form['user_id']      # 사용자 아이디
password = request.form['password']    # 비밀번호
nickname = request.form['nickname']    # 닉네임

# 비밀번호를 해싱하여 보안 강화 (데이터베이스에는 해시값 저장)
hashed_pw = generate_password_hash(password)

# UUID를 사용해 사용자 고유 ID 생성 (중복 없는 유일한 식별자)
user_uuid = str(uuid.uuid4())

try:
    # SQL 쿼리문 작성 - 사용자를 users 테이블에 삽입
    query = "INSERT INTO users (id, user_id, password, nickname) VALUES ("

    # 커서를 통해 SQL 실행. 각 변수는 튜플로 전달되어 보안상 안전하게 처리됨 (SQL
    cursor.execute(query, (user_uuid, user_id, hashed_pw, nickname))

    # 변경사항을 데이터베이스에 반영 (커밋)
    db.commit()

    # 회원가입 성공 메시지와 함께 생성된 UUID를 출력
    return f"회원가입 완료! UUID: {user_uuid}"

except mysql.connector.IntegrityError:
    # user_id가 UNIQUE 제약조건에 위배될 경우 (이미 존재하는 ID)
    return "이미 존재하는 아이디입니다."

# 이 파일이 메인 모듈로 실행될 경우 Flask 서버 실행 (디버그 모드 켜짐)
if __name__ == '__main__':
    app.run(debug=True)

```

실행 결과_성공 시에는 아래와 같이 로컬 서버: 127.0.0.1:5000 이 열리며 해당 서버 내부에
서 웹 사이트를 통해 HTTP 통신을 통한 데이터 전송 및 받아 읽기를 할 수 있게 됩니다.

```
문제   출력   디버그 콘솔   터미널   포트
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\정동진\Desktop\씬 스토리 맵 웹\회원가입 기능> cmd /C "c:\Python312\python.exe c:\Users\정동진\.vscode\extensions\
s\debugpy\launcher 51908 -- "c:\Users\정동진\Desktop\씬 스토리 맵 웹\회원가입 기능\app.py" "
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 468-813-383
```

<3> 만약 위의 결과가 나오지 않는다면 윈도우 터미널을 실행해 다음의 코드를 그대로 입력해 한 줄 한 줄 실행하면 성공적으로 될 겁니다.

단, 이때 아래 코드 중의 'your_password' 에는 여러분의 mysql 루트 계정의 비밀번호를 입력해야 합니다.

1. MySQL 루트 계정으로 로그인

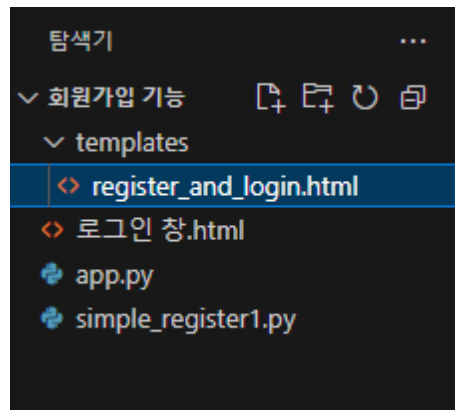
```
mysql -u root -p
```

2. 사용자 재설정 또는 권한 재부여

```
-- 자신의 mysql 비밀번호를 사용하는 루트 계정으로써 사용자 재설정하기.
ALTER USER 'scene_story_map'@'localhost' IDENTIFIED BY 'your_mysql_password';

-- 이후, 접근 가능한 모든 권한을 재부여해서 어떤 웹 서버 프로그램에서도 접근가능하게 만들기.
GRANT ALL PRIVILEGES ON userdb.* TO 'scene_story_map'@'localhost';
FLUSH PRIVILEGES;
```

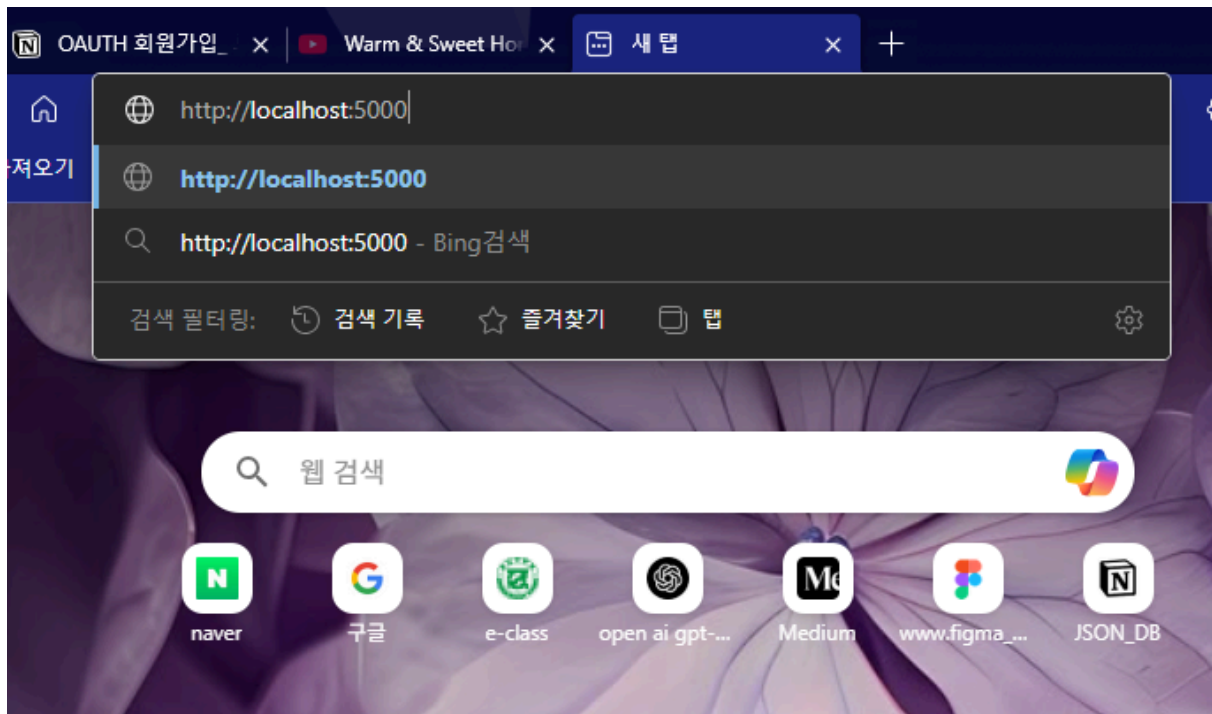
다음으로 자신의 `app.py` 파일이 열린 visual studio code 편집기로 이동해서 아래 사진처럼 templates 폴더를 만든 다음, 해당 폴더 내부에 우리가 방금 작성했던 `register_and_login.html` 파일을 넣어 줍니다.



이렇게 하는 이유는 아래 사진처럼 `app.py` 의 `index` 함수에서 리턴하는 회원가입 및 로그인 html 파일이 `render_template` 이라는 라이브러리 함수를 통해 반환되게끔 명시되어 있기 때문입니다. 즉, `index` 함수가 **templates** 내부 폴더에 저장된 html 파일만을 읽어들이게 정의되어 있기 때문입니다.

```
17 @app.route('/')
18 def index():
19     return render_template('register_and_login.html')
20
```

<4> 브라우저에서 아래 사진처럼 <http://localhost:5000> 을 통해서 접속하면 해당 로컬 서버에 자신의 html 파일이 입력을 받아 `app.py` 로 전송할 입력 데이터 값을 기다리고 있게 됩니다.

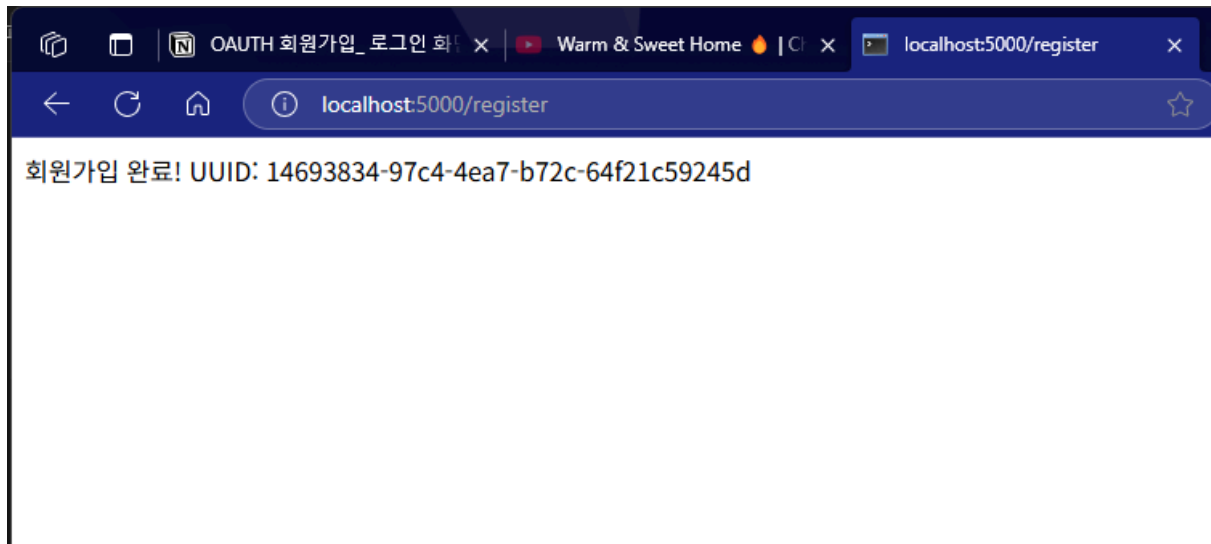


 A screenshot of a web page titled 'Register'. The page has a dark blue header with navigation icons (back, refresh, home) and the address 'localhost:5000'. The main content area is white and contains the title 'Register' in bold. Below the title are three input fields labeled '아이디:' (ID), '비밀번호:' (Password), and '닉네임:' (Nickname). At the bottom, there is a button labeled '회원가입' (Sign Up).

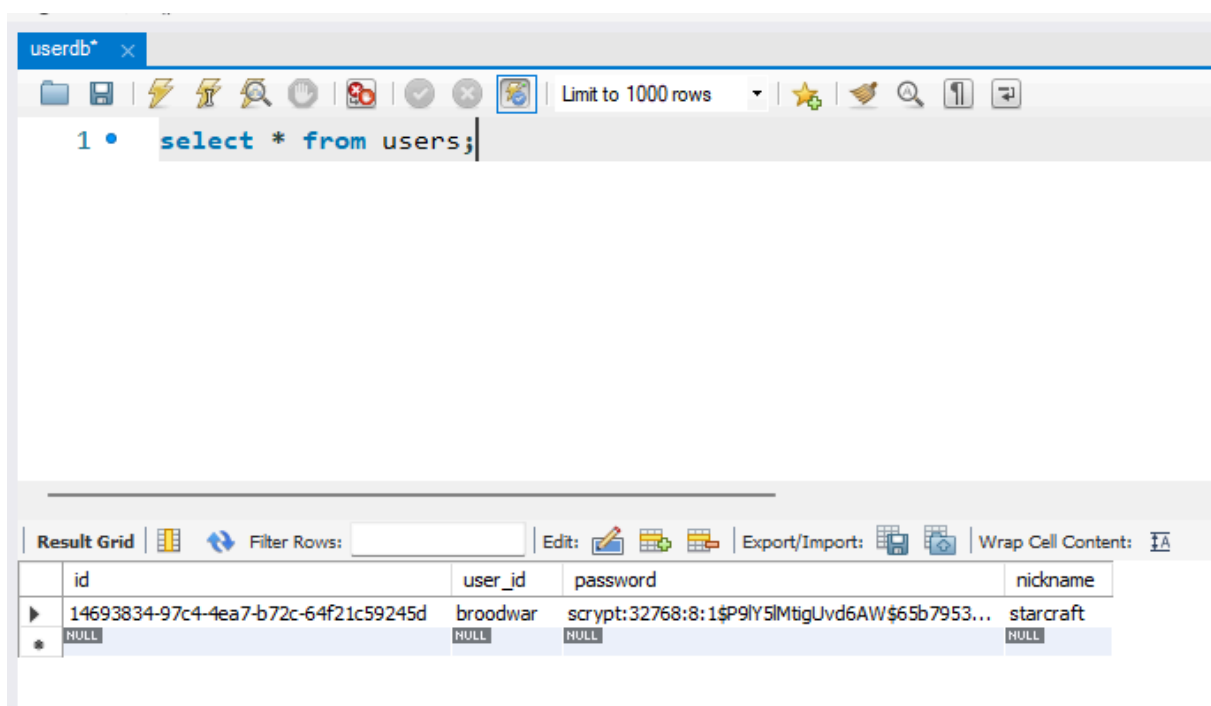
<5> 이제 원하는 종류의 아이디와 비밀번호, 닉네임을 입력한 후 회원가입 버튼을 클릭합니다.

성공 시 아래와 같이 회원가입에 성공했다는 메시지를 출력할 겁니다.

```
try:
    cursor.execute(
        "INSERT INTO users (id, user_id, password, nickname) VALUES (%s, %s, %s, %s)",
        (user_uuid, user_id, hashed_pw, nickname)
    )
    db.commit()
    return f"회원가입 완료! UUID: {user_uuid}"
```



<6> mysql workbench 의 쿼리문 입력 창에서 아래와 같은 쿼리문을 입력하고 실행해 users 테이블에 입력한 회원의 정보가 저장되어 있는지 확인합니다.



비밀번호로는 zergling 을 입력했는데 위 사진에서 보다시피 비밀번호는 아래의 'generate_password_hash' 함수에 의해 복호화된 script 파일로 저장되어 있습니다.

```
def register():
    user_id = request.form['user_id']
    password = request.form['password']
    nickname = request.form['nickname']

    hashed_pw = generate_password_hash(password)
```

이 함수는 werkzeug.security 라이브러리에서 온 친구입니다.

```
4 from werkzeug.security import generate_password_hash
```

또 1469..5d 값으로 저장되어 있던 id 값은 아래와 같이 uuid 라이브러리의 uuid4 함수를 통해 고유한 id 값을 자동 생성한 결과입니다.

```
user_uuid = str(uuid.uuid4())
```

2.1_2. 간단한 로그인 기능 구현하기. with flask 서버, mysql, html 코드.

<1> visual studio code 에 templates 폴더를 만든 다음, 해당 폴더에 다음의 로그인 화면을 위한 html 코드 파일을 만들어 넣습니다.

```
app_login.py x login.html x
templates > login.html > html > body > div.container > form
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4     <meta charset="UTF-8">
5     <title>로그인</title>
6
7     <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
8 </head>
9
10 <body>
11     <div class="container">
12         <h2>로그인</h2>
13
14         <form action="/login" method="post">
15             <input type="text" name="user_id" placeholder="아이디" required><br>
16             <input type="password" name="password" placeholder="비밀번호" required><br>
17             <button type="submit">로그인</button>
18         </form>
19     </div>
20 </body>
21 </html>
```

<2> templates 폴더에 다음의 초간단 프로필 화면을 위한 html 코드 파일을 만들어 넣습니다.

```
app_login.py x profile.html x
templates > profile.html > ...
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4     <meta charset="UTF-8">
5     <title>프로필</title>
6 </head>
7 <body>
8     <h2>환영합니다, {{ nickname }}님!</h2>
9 </body>
10 </html>
11
```

<3> static 폴더를 만든 다음, 다음의 간단한 css 파일을 만들어 넣습니다.

```

static > # style.css > .container
1  body {
2      font-family: 'Arial';
3      background-color: #f0f0f0;
4      text-align: center;
5      margin-top: 100px;
6  }
7
8  .container {
9      background-color: #fff;
10     padding: 30px;
11     display: inline-block;
12     border-radius: 10px;
13     box-shadow: 0 0 10px rgba(0,0,0,0.1);
14 }
15

```

<4> 다음의 app_login.py 파일을 작성해 동일한 프로젝트 폴더에 담아둡니다.

```

from flask import Flask, render_template, request, redirect, url_for
import mysql.connector
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__) # ← Flask 가 로컬서버에서 실행할 application program 을

# DB 연결 정보
db = mysql.connector.connect(
    host="127.0.0.1", # ← 당신의 MySQL DB 의 host 주소 값(connection 정보에서 host)
    user="root",      # ← 당신의 MySQL 사용자명
    password="*****", # ← 당신의 MySQL 비밀번호
    database="userdb" # ← 정보를 저장할 MySQL DB 의 이름
)

# 데이터베이스에서 딕셔너리 형태로 결과를 받기 위해 커서 객체를 생성했습니다.
cursor = db.cursor(dictionary=True) # 결과를 {column_name: value} 형태로 반환

```

```

# 루트 URL ('/')로 접속했을 때 실행되는 뷰 함수 index 를 정의합니다.
@app.route('/')
def index():
    # 클라이언트에게 'login.html' 템플릿을 렌더링하여 응답합니다.
    return render_template('login.html')

# '/login' 경로로 POST 요청이 왔을 때 실행되는 뷰 함수, login 함수를 정의합니다.
@app.route('/login', methods=['POST'])
def login():
    # 로그인 폼에서 입력한 사용자 아이디 값을 가져옵니다.
    user_id = request.form['user_id']

    # 로그인 폼에서 입력한 비밀번호 값을 가져옴
    password = request.form['password']

    # 데이터베이스에서 해당 user_id를 가진 유저 정보를 가져옴
    cursor.execute("SELECT * FROM users WHERE user_id = %s", (user_id,))

    # SQL 쿼리 결과 중 첫 번째 행을 딕셔너리 형태로 가져옴 (없으면 None 반환)
    user = cursor.fetchone()

    # 유저 정보가 존재할 경우
    if user:
        # 데이터베이스에 저장된 해시된 비밀번호 값을 변수에 저장
        stored_password_hash = user['password']

        # 입력된 비밀번호와 저장된 해시 값이 일치하는지 확인
        if check_password_hash(stored_password_hash, password):
            # 비밀번호가 일치하면 로그인 성공 → 'profile.html' 렌더링, 닉네임을 함께 전달
            return render_template("profile.html", nickname=user["nickname"])
        else:
            # 비밀번호가 틀렸을 경우 에러 메시지 반환
            return "❌ 입력하신 비밀번호가 틀렸습니다."
    else:
        # 해당 user_id를 가진 유저가 없을 경우 에러 메시지 반환
        return "❌ 해당 아이디는 존재하지 않습니다."

```

```
# 이 파일이 메인 프로그램으로 인해 실행될 때 웹 서버 app 실행 (디버그 모드 켜짐)
if __name__ == '__main__':
    app.run(debug=True)
```

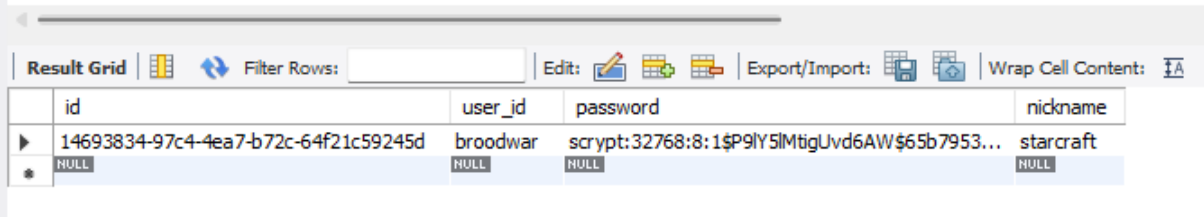
<5> 저희는 앞서 mysql 에 아래와 같은 테이블을 이미 동일한 users 이름으로 생성해 두었으니 해당 테이블을 수정하지 말고 그대로 둡니다.

```
CREATE DATABASE IF NOT EXISTS user_db;

USE userdb;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id VARCHAR(500) NOT NULL UNIQUE,
    password VARCHAR(500) NOT NULL,
    nickname VARCHAR(500) NOT NULL
);
```

저희가 앞서 만든 회원으로 인해 이미 1개의 데이터 객체가 생성되어 있는 걸 확인합니다. 작성할 sql 쿼리문은 'select * from users;' 입니다.



The screenshot shows a database query result grid with the following data:

	id	user_id	password	nickname
▶	14693834-97c4-4ea7-b72c-64f21c59245d	broodwar	scrypt:32768:8:1\$P9lY5lMtigUvd6AW\$65b7953...	starcraft
*	NULL	NULL	NULL	NULL

<6> app_login.py 를 실행합니다.

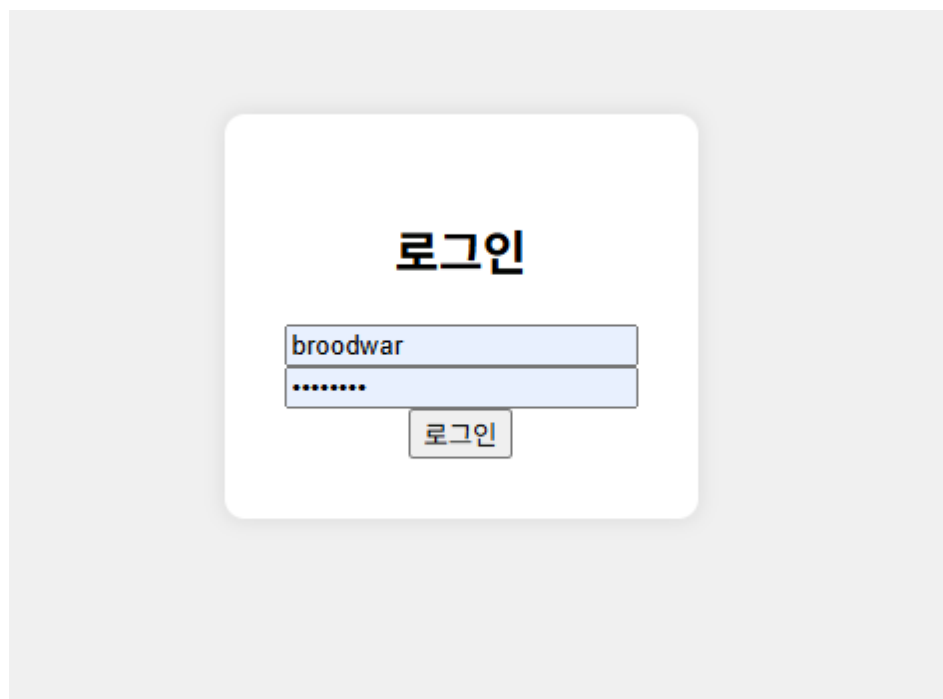
```

c:\Users\정동진\Desktop\썬 스토리 맵 웹\회원가입 기능>
c:\Users\정동진\Desktop\썬 스토리 맵 웹\회원가입 기능> c: && cd "c:\Users\정동진\Desktop\썬 스토리 맵 웹\회원가입 기능" &
.\vscode\extensions\ms-python.debugpy-2025.6.0-win32-x64\bundle\libs\debugpy\launcher 58459 -- "c:\Users\정동진\Desktop\
* Serving Flask app 'app_login'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 468-813-383

```

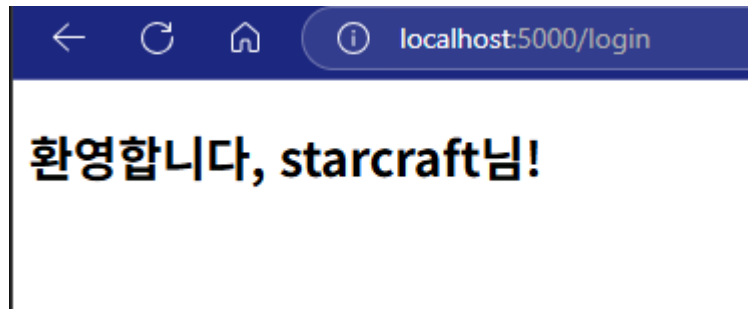
위 사진에서 보드시피 앞선 회원가입에서의 로컬 서버와 동일한 5000 번 포트의 localhost 서버에서 위의 파이썬 코드의 결과가 실행되고 있음을 알 수 있습니다.

<7> http://localhost:5000 번 대의 서버로 접속해서 자신의 웹 사이트 출력 결과를 확인 합니다.



자신이 저장해 두었던 회원 계정의 id 값과 비밀번호 값을 입력한 다음 '로그인' 버튼을 클릭 해 로그인을 시도합니다.

<8> 다음의 profile.html 파일이 등장했다면 성공입니다! 축하합니다!



2.1.3. \ 프로필 사진의 변경 결과와 프로필 소개문구 변경 결과가 내 계정의 프로필에 영구적으로 데이터 베이스를 통해서 반영되게 하는 방법?

<1> profile.html 파일에서 프로필 사진의 파일을 삽입해서 유저 개체의 데이터 베이스, userdb 에 자신만의 프로필 사진을 저장하게 하기.

1단계_ 먼저 아래의 파이썬 코드를 복사해서 여러분만의 코드로써, 비주얼 스튜디오에 붙여넣으세요 :)

단, 파일의 이름은 **flask_profile_pic_for_DB.py** 로 해 주세요 😊

각 코드가 어떤 역할을 수행하고 있는지 자세한 주석문을 달았으니 처음부터 끝까지 한 번 읽어 보시면 코드의 행동을 이해하는데 큰 도움이 될 겁니다 😊

```
# 필요한 라이브러리들을 불러오기
from flask import Flask, request, redirect, url_for, render_template, session
from werkzeug.utils import secure_filename # 파일 이름을 안전하게 다루기 위한 !
import os # 파일 경로 등을 다루기 위한 표준 모듈
import uuid # 고유한 ID를 만들기 위한 모듈
import pymysql # MySQL 데이터베이스에 접속하기 위한 라이브러리 (for 일부 작업)
from werkzeug.security import generate_password_hash, check_password_hash
import mysql.connector # MySQL 데이터베이스 접속을 위한 또 다른 라이브러리 (머

# Flask 앱 객체 생성
app = Flask(__name__)
```

```

app.secret_key = 'your_secret_key' # 세션에 필요한 비밀키 설정

# 프로필 사진 저장 폴더 설정
UPLOAD_FOLDER = 'static\\profile_pics'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER # Flask 설정에 저장

# MySQL 데이터베이스 연결 설정
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="*****",
    database="userdb"
)
# 데이터베이스 커서 생성 (dictionary=True 옵션: 결과를 딕셔너리 형태로 반환)
cursor = db.cursor(dictionary=True)

# 기본 페이지 ('/') : 회원가입 및 로그인 페이지 렌더링
@app.route('/')
def index():
    return render_template('register_and_login.html')

# 회원가입 처리 라우트
@app.route('/register', methods=['POST'])
def register():
    # 폼으로부터 입력된 데이터 가져오기
    user_id = request.form['user_id']
    password = request.form['password']
    nickname = request.form['nickname']

    # 비밀번호를 해싱하여 저장
    hashed_pw = generate_password_hash(password)
    # 사용자 고유 UUID 생성
    user_uuid = str(uuid.uuid4())

    try:
        # users 테이블에 새 사용자 정보 삽입
        cursor.execute(
            "INSERT INTO users (id, user_id, password, nickname) VALUES (%s, %s, %s, %s)"

```

```

        (user_uuid, user_id, hashed_pw, nickname)
    )
    db.commit() # 변경사항 저장
    return redirect(url_for('login')) # 회원가입 성공 후 로그인 페이지로 이동
except mysql.connector.IntegrityError:
    # 이미 존재하는 아이디일 경우 예외 처리
    return "이미 존재하는 아이디입니다."

# 로그인 처리 라우트
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        # 폼에서 입력된 아이디와 비밀번호 가져오기
        user_id = request.form['user_id']
        password = request.form['password']

        # 입력된 아이디로 사용자 정보 조회
        cursor.execute("SELECT * FROM users WHERE user_id = %s", (user_id,))
        user = cursor.fetchone()

        if user:
            stored_password_hash = user['password']
            # 저장된 해시 비밀번호와 입력 비밀번호 비교
            if check_password_hash(stored_password_hash, password):
                # 비밀번호 일치 시 세션에 사용자 ID 저장
                session['user_id'] = user['id']
                return redirect(url_for('profile')) # 프로필 페이지로 이동
            else:
                return "❌ 비밀번호가 틀렸습니다."
        else:
            return "❌ 아이디가 존재하지 않습니다."
    return render_template('login.html') # GET 요청 시 로그인 페이지 렌더링

# 별도로 DB 연결을 새로 생성하는 함수 (파일 업로드 등에서 사용)
def get_db_connection():
    return pymysql.connect(
        host='localhost',
        user='root',

```

```

        password='frziu2357*',
        db='userdb',
        charset='utf8mb4',
        cursorclass=pymysql.cursors.DictCursor
    )

# 프로필 사진 업로드 처리 라우트
@app.route('/upload_profile_pic', methods=['POST'])
def upload_profile_pic():
    if 'user_id' not in session:
        return redirect('/login') # 로그인하지 않은 경우 로그인 페이지로 이동

    file = request.files['profile_pic'] # 업로드된 파일 가져오기
    if file and file.filename != '': # 파일이 정상적으로 존재하는지 확인
        filename = secure_filename(file.filename) # 파일명을 안전하게 변환
        ext = os.path.splitext(filename)[1] # 파일 확장자 추출
        unique_name = str(uuid.uuid4()) + ext # 고유한 파일명 생성
        save_path = os.path.join(app.config['UPLOAD_FOLDER'], unique_name)
        file.save(save_path) # 서버에 파일 저장
        profile_img_web_path = f"profile_pics/{unique_name}" # 웹에서 접근할 경:

    # 새로 만든 DB 연결 사용
    conn = get_db_connection()
    with conn.cursor() as cursor:
        # 해당 사용자의 profile_img_url 업데이트
        sql = "UPDATE users SET profile_img_url = %s WHERE id = %s"
        cursor.execute(sql, (save_path, session['user_id']))
        conn.commit() # 변경사항 저장
    conn.close() # DB 연결 종료

    return redirect(url_for('profile')) # 파일 업로드 후 프로필 페이지로 이동

# 프로필 페이지 라우트
@app.route('/profile')
def profile():
    if 'user_id' not in session:
        return redirect('/login') # 로그인하지 않은 경우 로그인 페이지로 이동

```

```

# 새로 만든 DB 연결 사용
conn = get_db_connection()
with conn.cursor() as cursor:
    # 현재 로그인한 사용자의 닉네임과 프로필 이미지 경로 가져오기
    sql = "SELECT nickname, profile_img_url FROM users WHERE id = %s"
    cursor.execute(sql, (session['user_id'],))
    user = cursor.fetchone()
conn.close()

if user['profile_img_url']:
    # 저장된 경로에서 'static/' 제거하고 슬래시를 일관되게 수정
    profile_img_url = user['profile_img_url'].replace('static/', '').replace('\\', '/')
else:
    # 프로필 사진이 없는 경우 기본 이미지 사용
    profile_img_url = 'static\\profile_pics\\a7a4c0ca-ed78-4d12-91e7-7c7cb5

# 프로필 템플릿 렌더링 (닉네임과 프로필 이미지 경로 전달)
return render_template('profile.html', nickname=user['nickname'], profile_in

# 앱 실행 (디버그 모드로 실행하여 수정사항 즉시 반영)
if __name__ == '__main__':
    app.run(debug=True)

```

2단계_ **templates** 라는 폴더를 생성한 후, 아래의 각 html 코드를 '파일 이름을 동일하게' 작성해서 3개 html 파일 모두를 해당 폴더에 삽입하세요 😊

1: **login.html** 파일의 코드.

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>로그인</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

```

```

</head>

<body>
  <div class="container">
    <h2>로그인</h2>

    <form action="/login" method="post">
      <input type="text" name="user_id" placeholder="아이디" required><br>
      <input type="password" name="password" placeholder="비밀번호" required><br>
      <button type="submit">로그인</button>
    </form>
  </div>
</body>
</html>

```

2: profile.html 파일의 코드.

```

<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="UTF-8">
  <title>프로필</title>
  <style>
    .profile-container {
      display: flex;
      flex-direction: column;
      align-items: center;
      margin-top: 50px;
    }

    .profile-pic {
      width: 150px;
      height: 150px;
      border-radius: 50%;
      background-color: #ccc;
      background-size: cover;
    }
  </style>

```

```

        background-position: center;
        cursor: pointer;
        transition: box-shadow 0.3s;
    }

    .profile-pic:hover {
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
    }

    #fileInput {
        display: none;
    }

    h2 {
        text-align: center;
    }
</style>
</head>

<body>
    <h2>환영합니다, {{ nickname }}님!</h2>

    <form action="{{ url_for('upload_profile_pic') }}" method="POST" enctype=

        <label for="fileInput">
            <div class="profile-pic" id="profilePic"
                style="background-image: url('{{ profile_img_url }}');"></div>
            </label>

            <input type="file" id="fileInput" name="profile_pic" accept="image/png,
            <button type="submit">프로필 사진 저장</button>

        </form>

    <script>
        function loadPreview(event) {
            const input = event.target;

```

```

        if (input.files && input.files[0]) {
            const reader = new FileReader();

            reader.onload = function (e) {
                document.getElementById('profilePic').style.backgroundImage = url;
            };
            reader.readAsDataURL(input.files[0]);
        }
    }
}
</script>

</body>

</html>

```

3: register_and_login.html 파일의 코드.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Register and Login</title>
</head>
<body>
    <h2>Register</h2>
    <form action="/register" method="POST">

        <label for="user_id">아이디:</label>
        <input type="text" name="user_id" required><br><br>

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

        <label for="nickname">닉네임:</label>
        <input type="text" name="nickname" required><br><br>

        <button type="submit">회원가입</button>
    </form>

```



```

</form>

<!-- 로그인 창으로 이동 버튼 추가 -->
<br>
<a href="{{ url_for('login') }}">
  <button>로그인 창으로 이동</button>
</a>
</body>
</html>

```

3단계_ 동일 프로젝트 폴더에 static 폴더를 생성하고, 해당 폴더 안에 profile_pics 폴더를 생성하세요 😊

다음으로 static 폴더 안에(profile_pics 폴더 안이 아닌) 아래와 같은 "style.css" 파일을 작성해 삽입하세요 😊 파일, 폴더 이름을 동일하게 설정하는 거 잊지 마세요 😊

profile_pics 폴더에는 여러분이 직접 프로필 사진으로 설정한 모든 사진들이 담길 겁니다.

```

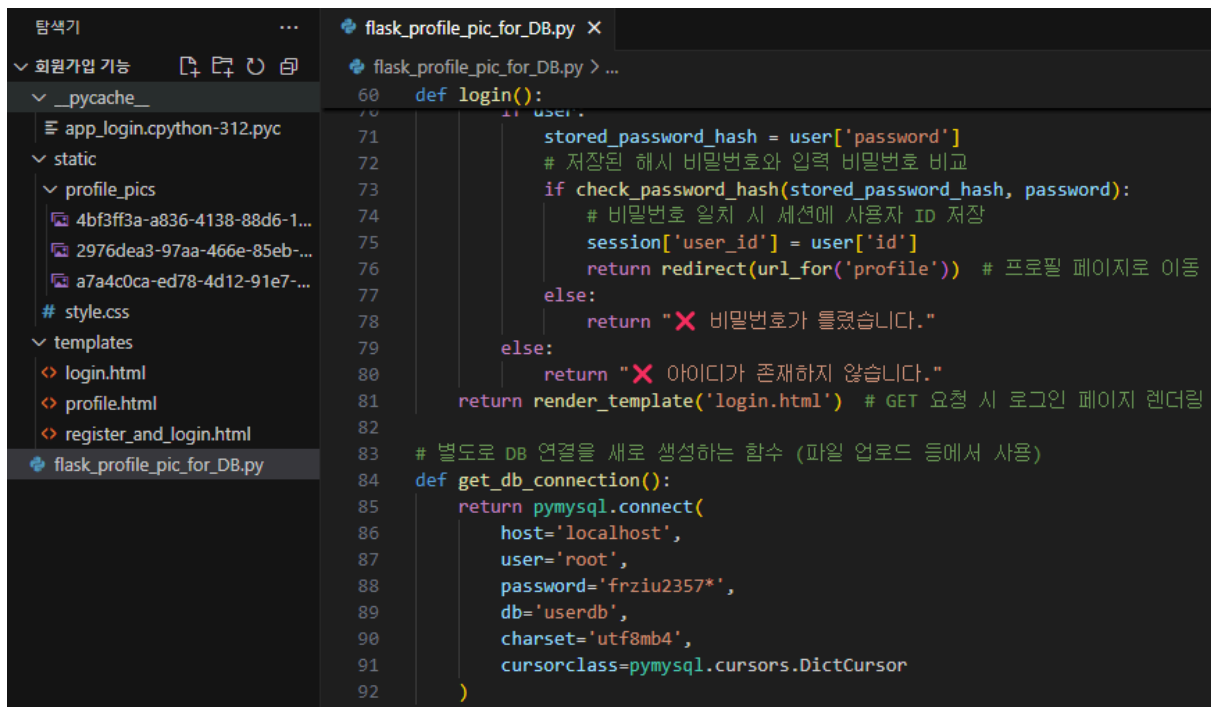
body {
  font-family: 'Arial';
  background-color: #f0f0f0;
  text-align: center;
  margin-top: 100px;
}

.container {
  background-color: #fff;
  padding: 30px;
  display: inline-block;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

```

4단계_ 같은 프로젝트 폴더에 __pycache__ 폴더를 생성하세요. 언더 바가 2개 연속으로 이어진 것이니 참고하세요 😊 (static 폴더 안이 아닙니다.)

그럼 아래와 같은 파일 구조로 구성된 프로젝트 폴더가 완성될 겁니다!



```
60 def login():
61     user = request.json
62     stored_password_hash = user['password']
63     # 저장된 해시 비밀번호와 입력 비밀번호 비교
64     if check_password_hash(stored_password_hash, password):
65         # 비밀번호 일치 시 세션에 사용자 ID 저장
66         session['user_id'] = user['id']
67         return redirect(url_for('profile')) # 프로필 페이지로 이동
68     else:
69         return "❌ 비밀번호가 틀렸습니다."
70
71 else:
72     return "❌ 아이디가 존재하지 않습니다."
73
74 return render_template('login.html') # GET 요청 시 로그인 페이지 렌더링
75
76 # 별도로 DB 연결을 새로 생성하는 함수 (파일 업로드 등에서 사용)
77 def get_db_connection():
78     return pymysql.connect(
79         host='localhost',
80         user='root',
81         password='frziu2357*',
82         db='userdb',
83         charset='utf8mb4',
84         cursorclass=pymysql.cursors.DictCursor
85     )
86
87
88
89
90
91
92
```

5단계_ flask_profile_pic_for_DB.py 파일을 ctrl+f5 키를 눌러서 디버거로써 실행하세요 😊

이어서 빈 웹 페이지를 연 다음, url 주소 창에 '127.0.0.1:5000' 을 입력한 다음 엔터 키를 눌러 flask 에 의해 활성화된 로컬 서버로 접속하세요 😊

아래는 flask 파일을 실행해 서버가 정상 작동할 때 비주얼 스튜디오의 터미널에서 보이는 상태 코드입니다.

```

C:\Users\정동진\Desktop\scene 웹 파일\회원가입 기능>
C:\Users\정동진\Desktop\scene 웹 파일\회원가입 기능> c: && cd "c:\Users\정동진\Desktop\scene 웹 파일\회원가입 기능" && c
\libs\debugpy\launcher 53957 -- "c:\Users\정동진\Desktop\scene 웹 파일\회원가입 기능\flask_profile_pic_for_DB.py" "
* Serving Flask app 'flask_profile_pic_for_DB'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 468-813-383

```

서버 파일 실행 후 127.0.0.1:5000 으로 접속한 결과입니다 😊

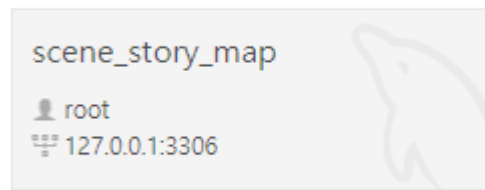
Register

아이디:

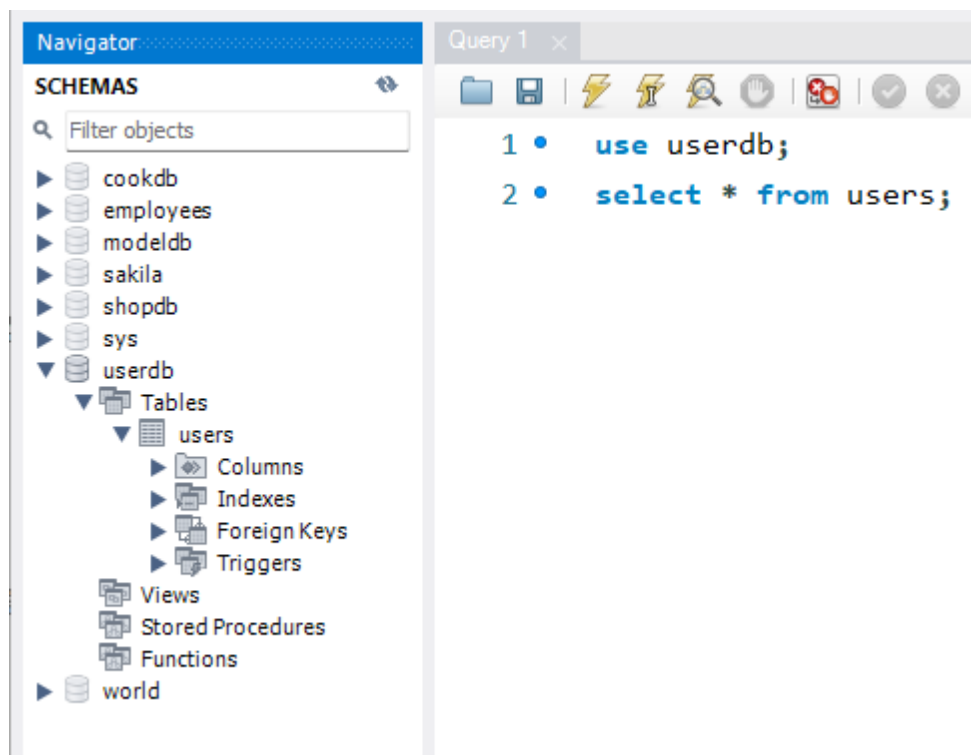
비밀번호:

닉네임:

6단계_ MySQL Workbench 프로그램을 실행한 다음 여러분이 만들어 둔 MySQL Connection 중 아래와 같은 scene_story_map connection 을 클릭해 접속합니다.



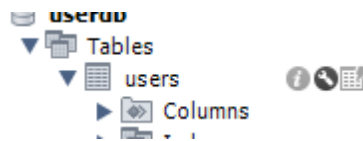
앞선 2_1_1. 단계에서 만들어 둔 users table 을 확인하고, 다음의 mysql 쿼리문을 작성한 후 번개 버튼을 눌러 쿼리문을 실행합니다 😊



실행 결과를 보시면 아래와 같이 가장 왼쪽에 처음 보는 profile_img_url 속성이 자신의 실행 결과에서는 보이지 않는 걸 확인할 수 있을 겁니다. 네, 이 속성은 지금 여러분의 users 테이블에 생성해야 하는 속성입니다 😊

id	user_id	password	nickname	profile_img_url
14693834-97c4-4ea7-b72c-64f21c59245d	broodwar	script:32768:8:1\$P9lY5lMtigUvd6AW\$65b7953...	starcraft	static/profile_pics/4bf3ff3a-a836-4138-88d6-18...
96bc660f-60ec-4935-a1dc-e2ea267326ac	prestine	script:32768:8:1\$Ie4ki2UitOcX8nO\$36f9c619...	prestine	NULL
NULL	NULL	NULL	NULL	NULL

users 테이블에 마우스를 올려 '수정' 아이콘을 아래와 같이 클릭하고,



아래와 같이, 'VARCHAR(255) 데이터 타입을 갖는, null 이 가능한 속성을 추가해 줍니다



Query 1 users - Table x

Table Name: Schema: **userdb**

Charset/Collation: Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
⚡ id	VARCHAR(300)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
◇ user_id	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ password	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ nickname	VARCHAR(500)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
◇ profile_img_url	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

workbench 프로그램을 그대로 열어둔 채로 다시 아까 접속했던 127.0.0.1:5000 웹 사이트의 화면으로 돌아옵니다. (이는 서버의 작용 결과를 즉시 확인하기 위함입니다 😊)

7단계_ 먼저 아래와 같은 회원가입 창에서 여러분만의 아이디, 비밀번호, 닉네임을 작성한 다음 회원가입 버튼을 클릭합니다.

작성하신 아이디, 비밀번호, 닉네임은 메모장에 적어두셔야 합니다. 비밀번호가 암호화될 것이기 때문에 적어두지 않고 잊어버리면 비밀번호를 다시 찾을 방법이 없습니다 😊

Register

아이디:

비밀번호:

닉네임:

회원가입

로그인 창으로 이동

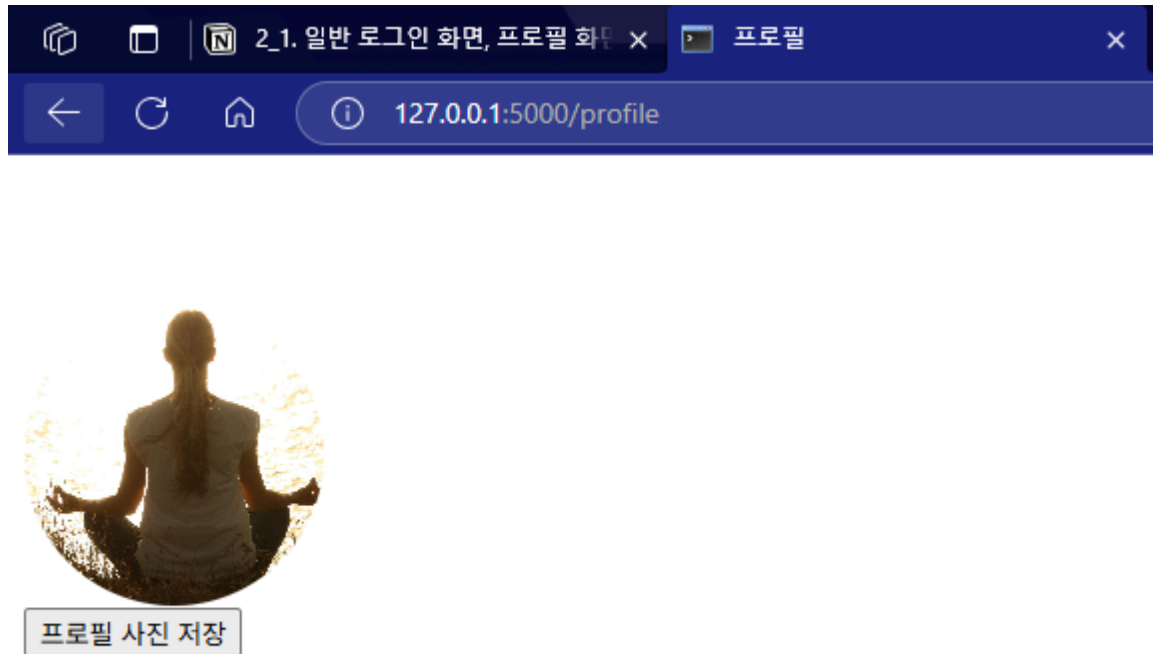
저는 이미 broodwar 아이디, starcraft 라는 닉네임, zergling 이라는 계정을 가입시켜 두었으므로 '로그인 창으로 이동' 버튼을 클릭했습니다 😊

8단계_ 아래와 같은 로그인 창에서 작성하셨던 아이디와 비밀번호를 작성한 후, 로그인 버튼을 클릭합니다 😊

로그인

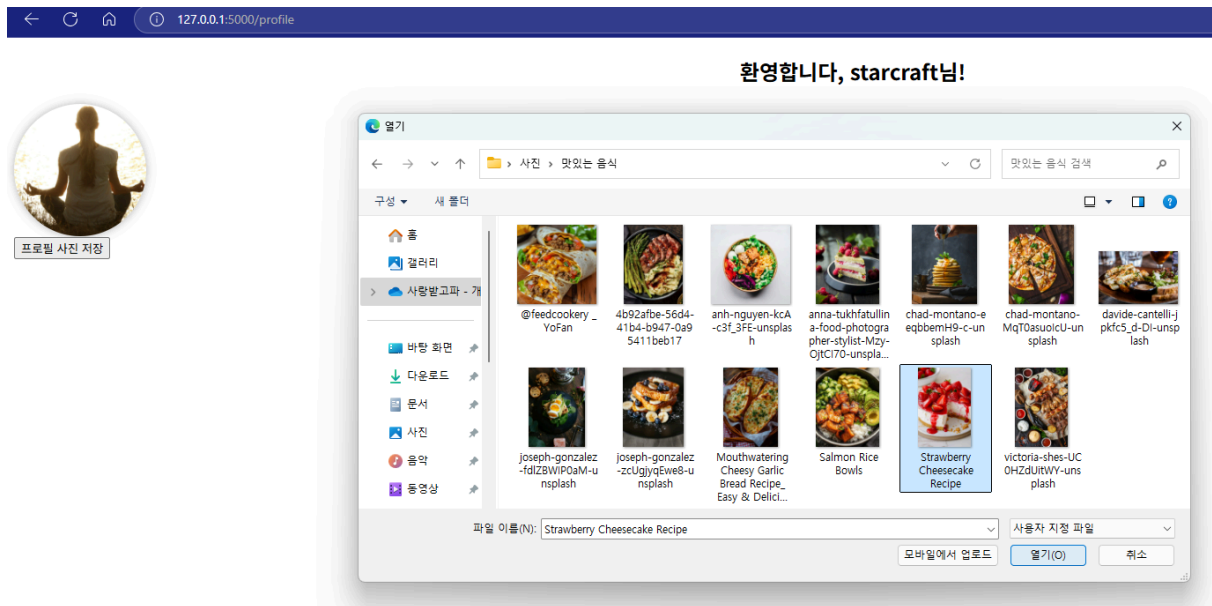
로그인

9단계_ 그러면 아래와 같은, 해당 계정의 프로필 화면이 등장할 것입니다. 동그라미로 표시된 프로필 사진 창을 클릭합니다.

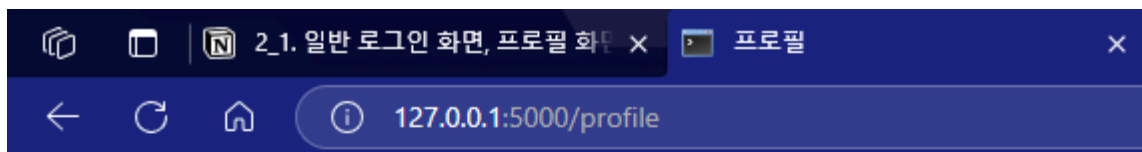


그러면 아래와 같이 여러분의 컴퓨터에서 여러분이 직접 하고 싶은 프로필 사진을 선택할 수 있게 됩니다.

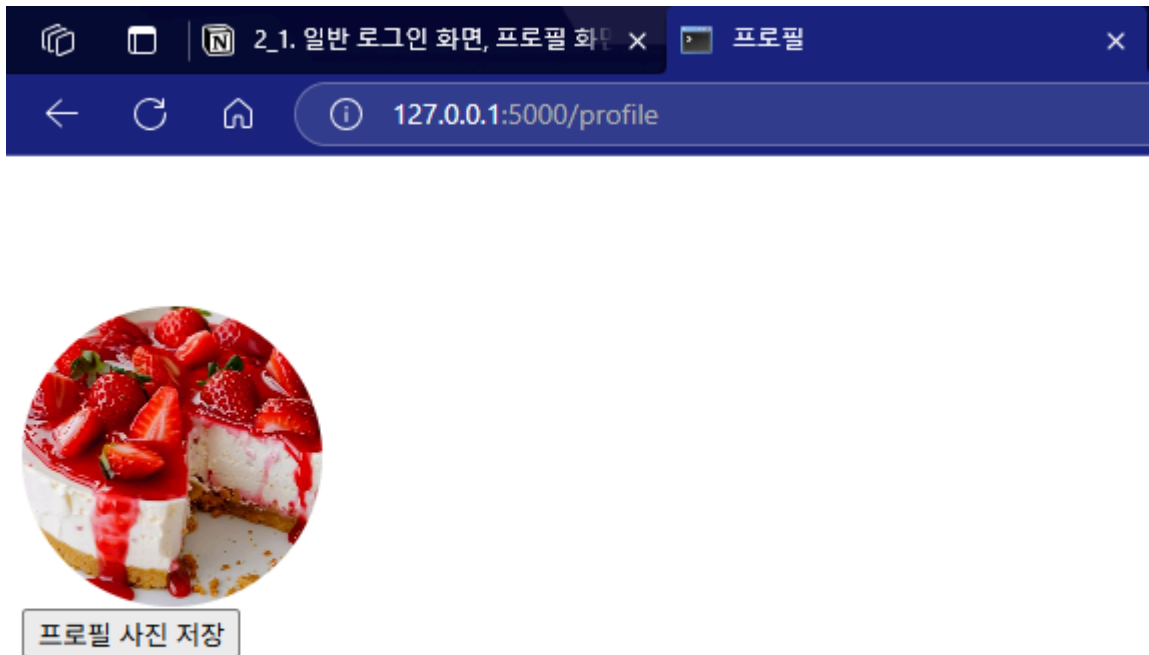
원하는 사진을 선택하고 '열기' 버튼을 누릅니다 😊



그러면 여전히 프로필 사진이 변경되지 않은 아래와 같은 창이 될 텐데, 당황하지 마시고 '프로필 사진 저장' 버튼을 클릭합니다 😊

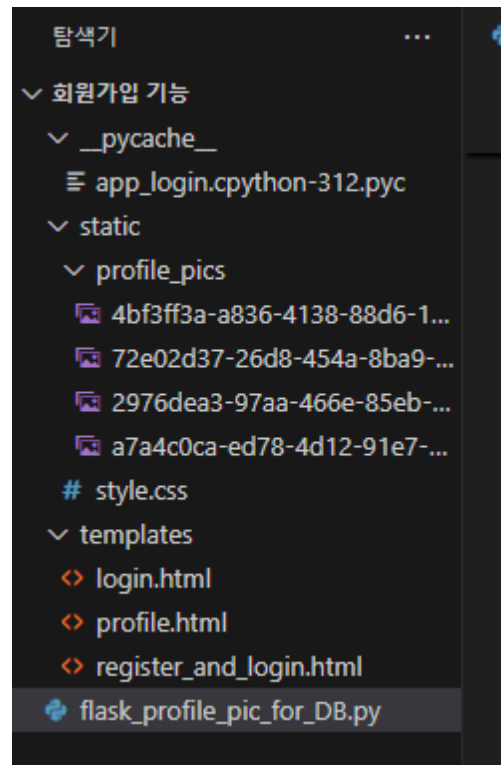


그러면 아래의 사진과 같이 내가 선택한 대로 나의 프로필 사진이 변경된 것을 잘 확인할 수 있습니다 😊

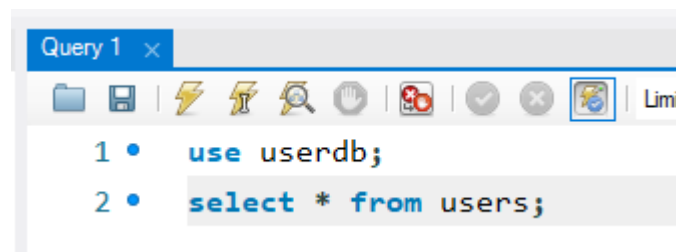


10단계_ 마지막으로 이 상태를 그대로 둔 채로 여러분의 비주얼 스튜디오 코드를 열어서 해당 `flask_profile_pic_for_DB.py` 파일을 담고 있던 프로젝트 폴더를 열어 봅니다.

그리고 static 폴더의 `profile_pics` 폴더를 열어 보면 아래와 같이 여러분이 선택한 사진 파일이 해당 폴더에 추가되어 있는 걸 확인할 수 있습니다.








이어서 아까 열어두었던 workbench 프로그램으로 돌아가서 아래와 같은 동일한 쿼리문을 작성하고 실행해 봅니다.



그러면 아래 사진에 보이는 것처럼, profile_img_url 속성에 해당 사진 파일의, 프로젝트 폴더에 대한 상대 경로가 들어가 있는 걸 확인할 수 있습니다 😊

데이터 베이스에 해당 사진 파일의 상대 경로가 새롭게 배정된 것입니다! 이는 이미 해당 계정에 프로필 사진이 설정된 이후라도 동일한 과정을 거쳐서 새로운 프로필 사진으로 등록이 가능합니다!

Result Grid					Filter Rows: <input type="text"/>		Edit:   		Export/Import:  		Wrap Cell Content: iA	
	id	user_id	password	nickname	profile_img_url							
▶	14693834-97c4-4ea7-b72c-64f21c59245d	broodwar	script:32768:8:1\$P9lY5lMtigUvd6AW\$65b7953...	starcraft	static/profile_pics/72e02d37-26d8-454a-8ba9-b...							
	96bc660f-60ec-4935-a1dc-e2ea267326ac	prestine	script:32768:8:1\$IEa4ki2UitOcX8nO\$36f9c619...	prestine	NULL							
*	NULL	NULL	NULL	NULL	NULL							

Wrap Cell Content: iA	
nickname	profile_img_url
starcraft	static/profile_pics/72e02d37-26d8-454a-8ba9-b...
prestine	NULL
NULL	NULL

<2> 프로필 소개문구를 작성할 텍스트 박스를 만들어 해당 박스 안에 입력되는 텍스트 값을
 유저 개체의 데이터 베이스, userdb 에 저장하게끔 데이터 베이스와 profile.html 코드를
 수정하기.