

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**ЗВІТ**

лабораторної работи №5

курсу «Основи WEB - технологій »

Тема: «**Створення клієнт-серверної системи авторизації сайту**»

Перевірив: Виконав:

Доц. Голубєв Л. П. Гр. ІП-01

Черпак Андрій

Київ 2023

**1. Завдання.**

Створити клієнт-серверну систему авторизації сайту, що складається з двох частин: Front-end частина – User-interface(JS/React): забезпечує введення логіну та паролю. Back-end частина (PHP/NodeJS): забезпечує введення, редагування та видалення логінів, паролів та статусу користувача (admin/user) та зберігає інформацію на сервері. Якщо введені дані збігаються, то видача інформації про користувача та його сторінки (дані з лаб. роб. №1), інакше – виведення повідомлення "Доступ заборонено".

**2. Хід роботи**

**Завдання 1.**

За допомогою текстового редактора JB Rider, було створено файл з кодом сервера з використанням Flask.py (index.py), допоміжний файл з функціями для роботи з базами даних (db\_utils.py), структуру WEB-сторінок реєстрації, авторизації та відображення профілів мовою HTML-5 (signup-form.html, login-form.html, profiles.html), каскадні таблиці стилів (common-styles.css, form-styles.css, profile-styles.css), а також JS-файли зі скриптами для веб-сторінок (log-in.js, sign-up.js, profiles.js, validations.js):

index.py:

from flask import Flask, request, Request, Response, make\_response, abort, jsonify, render\_template, redirect  
from hashlib import shake\_256, shake\_128  
from sqlite3 import connect  
from utils.db\_utils import (ensure\_created\_db, get\_user\_session, try\_login, check\_username, register\_new\_user,  
 get\_my\_profile, get\_all\_profiles, check\_for\_admin, update\_user, update\_passwd,   
 delete\_users\_profile, make\_admin)  
  
  
app = Flask(\_\_name\_\_)  
  
\_\_db\_name = 'users.db'  
cached\_access\_tokens: dict[int, dict[str, str | int]] = {}  
ensure\_created\_db(\_\_db\_name)  
  
  
def authenticate(user\_id: int, hashed\_passw: str) -> Response:  
 access\_token = shake\_128(hashed\_passw.encode()).hexdigest(5)  
 resp = make\_response(jsonify({"status": "Success", "user\_id": user\_id, "token": access\_token}), 200)  
 resp.set\_cookie("user\_id", str(user\_id))  
 resp.set\_cookie("token", access\_token)  
 return resp  
  
  
def check\_access(req: Request) -> bool:  
 token = req.cookies.get('token')  
 user\_id = req.cookies.get('user\_id', type=int)  
 if user\_id in cached\_access\_tokens:  
 if cached\_access\_tokens[user\_id]['ctr'] > 0 and cached\_access\_tokens[user\_id]['token'] == token:  
 cached\_access\_tokens[user\_id]['ctr'] -= 1  
 return True  
 else:  
 cached\_access\_tokens.pop(user\_id, 0)  
 else:  
 conn = connect(\_\_db\_name)  
 is\_correct = (token and user\_id and get\_user\_session(conn, user\_id) == token)  
 conn.close()  
 if is\_correct:  
 cached\_access\_tokens[user\_id] = {'token': token, 'ctr': 10}  
 return is\_correct  
  
  
@app.route('/')  
def index():  
 if check\_access(request):  
 return redirect("/profiles", code=302)  
 else:  
 return redirect("/login", code=302)  
  
  
@app.route('/login')  
def to\_login():  
 return render\_template('login-form.html')  
  
  
@app.route('/signup')  
def to\_signup():  
 return render\_template('signup-form.html')  
  
  
@app.route('/profiles')  
def to\_profiles():  
 return render\_template('profiles.html')  
  
  
@app.post('/api/log-in')  
def login():  
 hashed\_passw = shake\_256(request.form['passwd'].encode()).hexdigest(10)  
 conn = connect(\_\_db\_name)  
 user\_id = try\_login(conn, request.form['username'], hashed\_passw)  
 conn.close()  
 if user\_id:  
 return authenticate(user\_id, hashed\_passw)  
 else:  
 return jsonify({"status": "Failed"})  
  
  
@app.post('/api/sign-up')  
def signup():  
 conn = connect(\_\_db\_name)  
 username\_available = check\_username(conn, request.form['login'])  
 if username\_available:  
 hashed\_passw = shake\_256(request.form['passwd'].encode()).hexdigest(10)  
 user\_id = register\_new\_user(conn, request.form['login'], hashed\_passw, request.form['full\_name'],  
 int(request.form['id\_card']), request.form['faculty'], request.form['birthdate'],  
 request.form['address'])  
 conn.close()  
 return authenticate(user\_id, hashed\_passw)  
 else:  
 conn.close()  
 abort(400)  
  
  
@app.get('/api/sign-up/<string:username>')  
def check\_username\_availability(username: str):  
 conn = connect(\_\_db\_name)  
 username\_availability = check\_username(conn, username)  
 conn.close()  
 return jsonify({"available": username\_availability})  
  
  
@app.get('/api/my-profile')  
async def get\_my\_profile\_info():  
 if check\_access(request):  
 user\_id = request.cookies.get('user\_id', type=int)  
 conn = connect(\_\_db\_name)  
 profile: dict[str, str | int] = get\_my\_profile(conn, user\_id)  
 conn.close()  
 return jsonify(profile)  
 else:  
 abort(401)  
  
  
@app.put('/api/my-profile')  
async def update\_my\_profile\_info():  
 if check\_access(request):  
 conn = connect(\_\_db\_name)  
 user\_id = request.cookies.get('user\_id', type=int)  
 profile: dict[str, str | int] = get\_my\_profile(conn, user\_id)  
 username\_available = profile['login'] == request.form['login'] or check\_username(conn, request.form['login'])  
 if username\_available:  
 hashed\_passw = shake\_256(request.form['passwd'].encode()).hexdigest(10)  
 user\_id = update\_user(conn, user\_id, request.form['login'], request.form['full\_name'],  
 int(request.form['id\_card']), request.form['faculty'], request.form['birthdate'],  
 request.form['address'])  
  
 pwd = request.form['passwd'].encode()  
 if pwd is not None and len(pwd) > 0:  
 hashed\_passw = shake\_256(pwd).hexdigest(10)  
 update\_passwd(conn, user\_id, hashed\_passw)  
 conn.close()  
 return authenticate(user\_id, hashed\_passw)  
 else:  
 conn.close()  
 abort(400)  
 else:  
 abort(401)  
  
  
@app.get('/api/profiles')  
async def get\_profiles():  
 if check\_access(request):  
 user\_id = request.cookies.get('user\_id', type=int)  
 conn = connect(\_\_db\_name)  
 if check\_for\_admin(conn.cursor(), user\_id):  
 profiles: list[dict[str, str | int]] = get\_all\_profiles(conn)  
 conn.close()  
 return jsonify(profiles)  
 else:  
 conn.close()  
 abort(403)  
 else:  
 abort(401)  
  
  
@app.delete('/api/profiles/<int:deleted\_user\_id>')  
async def delete\_profile(deleted\_user\_id: int):  
 if check\_access(request):  
 user\_id = request.cookies.get('user\_id', type=int)  
 conn = connect(\_\_db\_name)  
 if check\_for\_admin(conn.cursor(), user\_id):  
 delete\_users\_profile(conn, deleted\_user\_id)  
 conn.close()  
 return jsonify({'status': 'Success'})  
 else:  
 conn.close()  
 abort(403)  
 else:  
 abort(401)  
  
  
@app.put('/api/profiles/<int:promoted\_user\_id>')  
async def make\_user\_admin(promoted\_user\_id: int):  
 if check\_access(request):  
 user\_id = request.cookies.get('user\_id', type=int)  
 conn = connect(\_\_db\_name)  
 if check\_for\_admin(conn.cursor(), user\_id):  
 make\_admin(conn, promoted\_user\_id)  
 conn.close()  
 return jsonify({'status': 'Success'})  
 else:  
 conn.close()  
 abort(403)  
 else:  
 abort(401)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run("0.0.0.0", port=3000, debug=False)

db\_utils.py:

from sqlite3 import Cursor, Connection, connect  
from hashlib import shake\_128  
  
  
def ensure\_created\_db(db\_name: str) -> None:  
 connection = connect(db\_name)  
 cursor: Cursor = connection.cursor()  
  
 cursor.execute('''  
 CREATE TABLE IF NOT EXISTS Users (  
 user\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 login TEXT(20) UNIQUE,  
 hashed\_password TEXT(20),  
 is\_admin TINYINT DEFAULT 0,  
 full\_name TEXT(50) NOT NULL,  
 id\_card INTEGER NOT NULL,  
 faculty TEXT(4) NOT NULL,  
 birthdate TEXT(10) NOT NULL,  
 address TEXT(50) NOT NULL  
 )  
 ''')  
  
 cursor.execute('CREATE INDEX IF NOT EXISTS idx\_user\_login ON Users (login)')  
  
 connection.commit()  
 connection.close()  
  
  
def check\_for\_admin(crs: Cursor, user\_id: int) -> bool:  
 crs.execute(f'SELECT is\_admin FROM Users WHERE user\_id = ?', (user\_id,))  
 return crs.fetchone()[0] == 1  
  
  
def register\_new\_user(connection: Connection, login: str, hashed\_passwd: str, full\_name: str, id\_card: int, faculty: str, birthdate: str, address: str) -> int:  
 crs: Cursor = connection.cursor()  
 crs.execute(f'INSERT INTO Users(login, hashed\_password, is\_admin, full\_name, id\_card, faculty, birthdate, address) VALUES (?, ?, 0, ?, ?, ?, ?, ?)',  
 (login, hashed\_passwd, full\_name, id\_card, faculty, birthdate, address))  
 connection.commit()  
  
 crs.execute(f'SELECT user\_id FROM Users WHERE login = ? AND hashed\_password = ?',  
 (login, hashed\_passwd))  
 return crs.fetchone()[0]  
  
  
def update\_user(connection: Connection, user\_id: int, login: str, full\_name: str, id\_card: int, faculty: str, birthdate: str, address: str) -> int:  
 crs: Cursor = connection.cursor()  
 crs.execute(f'UPDATE Users SET login = ?, full\_name = ?, id\_card = ?, faculty = ?, birthdate = ?, address = ? WHERE user\_id = ?',  
 (login, full\_name, id\_card, faculty, birthdate, address, user\_id))  
 connection.commit()  
  
 crs.execute(f'SELECT user\_id FROM Users WHERE login = ?',  
 (login, ))  
 return crs.fetchone()[0]  
  
  
def update\_passwd(connection: Connection, user\_id: int, hashed\_passwd: str) -> None:  
 crs: Cursor = connection.cursor()  
 crs.execute(f'UPDATE Users SET hashed\_password = ? WHERE user\_id = ?',  
 (hashed\_passwd, user\_id))  
 connection.commit()  
  
  
def try\_login(connection: Connection, login: str, hashed\_passwd: str) -> int | None:  
 crs: Cursor = connection.cursor()  
  
 crs.execute(f'SELECT user\_id FROM Users WHERE login = ? AND hashed\_password = ?',  
 (login, hashed\_passwd))  
 res = crs.fetchall()  
 return None if len(res) == 0 else res[0][0]  
  
  
def check\_username(connection: Connection, login: str) -> bool:  
 crs: Cursor = connection.cursor()  
  
 crs.execute(f'SELECT user\_id FROM Users WHERE login = ?',  
 (login,))  
 res = crs.fetchall()  
 return len(res) == 0  
  
  
def get\_user\_session(connection: Connection, user\_id: int) -> str|None:  
 crs: Cursor = connection.cursor()  
  
 crs.execute(f'SELECT hashed\_password FROM Users WHERE user\_id = ?',  
 (user\_id,))  
 res = crs.fetchall()  
 return shake\_128(res[0][0].encode()).hexdigest(5) if len(res) > 0 else None  
  
  
def make\_admin(connection: Connection, user\_id: int):  
 crs: Cursor = connection.cursor()  
 crs.execute(f'UPDATE Users SET is\_admin = 1 WHERE user\_id = ?', (user\_id,))  
 connection.commit()  
  
  
def get\_all\_profiles(connection: Connection) -> list[dict[str, str | int]]:  
 crs = connection.cursor()  
 crs.execute(  
 f'''SELECT user\_id, login, hashed\_password, is\_admin, full\_name, id\_card, faculty, birthdate, address  
 FROM Users  
 ORDER BY user\_id''')  
 return [{'user\_id': user\_id, 'login': login, 'hashed\_passwd': hashed\_password, 'is\_admin': is\_admin == 1,  
 'full\_name': full\_name, 'id\_card': id\_card, 'faculty': faculty, 'birthdate': birthdate, 'address': address}  
 for user\_id, login, hashed\_password, is\_admin, full\_name, id\_card, faculty, birthdate, address  
 in crs.fetchall()]  
  
  
def get\_my\_profile(connection: Connection, user\_id: int) -> dict[str, str | int]:  
 crs = connection.cursor()  
 crs.execute(  
 f'''SELECT user\_id, login, hashed\_password, is\_admin, full\_name, id\_card, faculty, birthdate, address  
 FROM Users  
 WHERE user\_id = ?  
 LIMIT 1''',  
 (user\_id,))  
  
 user\_id, login, hashed\_password, is\_admin, full\_name, id\_card, faculty, birthdate, address = crs.fetchone()  
 return {'user\_id': user\_id, 'login': login, 'is\_admin': is\_admin == 1,  
 'full\_name': full\_name, 'id\_card': id\_card, 'faculty': faculty, 'birthdate': birthdate, 'address': address}  
  
  
def delete\_users\_profile(connection: Connection, user\_id: int):  
 crs = connection.cursor()  
 crs.execute(  
 f'''DELETE FROM Users WHERE user\_id=?''', (user\_id,))  
 connection.commit()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ensure\_created\_db('users.db')

signup-form.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Sign Up</title>  
 <link rel="stylesheet" href="/static/common-styles.css">  
 <link rel="stylesheet" href="/static/form-styles.css">  
 <script src="https://unpkg.com/axios/dist/axios.min.js"></script>  
</head>  
<body>  
<div id="form-container">  
 <h1>Sign Up</h1>  
 <form id="form1">  
 <input type="text" id="username" name="username" placeholder="Username">  
 <input type="text" id="passwd" name="passwd" placeholder="Password">  
 </form>  
 <form id="form2" style="display: none">  
 <input type="text" id="full\_name" name="full\_name" placeholder="Full name: Черпак А. В.">  
 <input type="text" id="id\_card" name="id\_card" placeholder="ID-card: 987654321">  
 <input type="text" id="faculty" name="faculty" placeholder="Faculty: ФІОТ">  
 <input type="text" id="birthdate" name="birthdate" placeholder="Birthdate: 2003-10-15">  
 <input type="text" id="address" name="address" placeholder="Address: м. Житомир">  
 </form>  
 <button id="submit-button">Next</button>  
 <div>or <a href="/login">log in</a></div>  
</div>  
</body>  
<script src="/static/validations.js"></script>  
<script src="/static/sign-up.js"></script>  
</html>

login-form.html:

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>Log In</title>  
 <link rel="stylesheet" href="/static/common-styles.css">  
 <link rel="stylesheet" href="/static/form-styles.css">  
 <script src="https://unpkg.com/axios/dist/axios.min.js"></script>  
 </head>  
 <body>  
 <div id="form-container">  
 <h1>Log In</h1>  
 <form>  
 <input type="text" id="username" name="username" placeholder="Enter your username">  
 <input type="password" id="passwd" name="passwd" placeholder="Enter your password">  
 </form>  
 <button id="submit-button">Log In</button>  
 <div>or <a href="/signup">sign up</a></div>  
 </div>  
 </body>  
 <script src="/static/log-in.js"></script>  
</html>

profiles.html:

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <title>Profiles</title>  
 <link rel="stylesheet" href="/static/common-styles.css">  
 <link rel="stylesheet" href="/static/profile-styles.css">  
 <script src="https://unpkg.com/axios/dist/axios.min.js"></script>  
 </head>  
 <body>  
 <div id="profile-container">  
 <h1>Your profile:</h1>  
 <div class="data-element-container">  
 <div class="data-element-name">Login:</div>  
 <input type="text" id="username" name="username" placeholder="Username">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">Password:</div>  
 <input type="text" id="passwd" name="passwd" placeholder="Password">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">Full name:</div>  
 <input type="text" id="full\_name" name="full\_name" placeholder="Черпак А. В.">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">ID-card:</div>  
 <input type="text" id="id\_card" name="id\_card" placeholder="987654321">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">Faculty:</div>  
 <input type="text" id="faculty" name="faculty" placeholder="ФІОТ">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">Birthdate:</div>  
 <input type="text" id="birthdate" name="birthdate" placeholder="2003-10-15">  
 </div>  
 <div class="data-element-container">  
 <div class="data-element-name">Address:</div>  
 <input type="text" id="address" name="address" placeholder="м. Житомир">  
 </div>  
 <div class="data-element-container">  
 <button id="submit-button">Update</button>  
 </div>  
 </div>  
 <div id="profiles-container">  
 </div>  
 </body>  
 <script src="/static/validations.js"></script>  
 <script src="/static/profiles.js"></script>  
</html>

common-styles.css:

:*root* {  
 --main-bg-color: #052b54  
}  
  
html, body{  
 width: 100%;  
 height: 100%;  
 box-sizing: border-box;  
 background-color: var(--main-bg-color);  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 margin: 0;  
}

form-styles.css:

:*root* {  
 --form-bg-color: #043c75;  
 --form-text-color: #f1f1f1;  
 --input-text-color: #333;  
 --input-text-incorrect-color: #f33;  
 --submit-button-color: #0968a2;  
 --submit-button-hover-color: #2080b0;  
 --submit-button-incorrect-color: #f33;  
 --submit-button-disabled-color: #555;  
 --inputs-incorrect-bg-color: #ffb0d0;  
 --inputs-bg-color: #f1f1f1;  
}  
  
#form-container{  
 box-sizing: border-box;  
 background-color: var(--form-bg-color);  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding: 1%;  
 border-radius: 15px;  
 color: var(--form-text-color);  
 font-family: Arial, Helvetica, sans-serif;  
 font-size: 1.2em;  
}  
  
#form-container h1{  
 box-sizing: border-box;  
 text-align: center;  
 margin: 15px;  
 font-size: 2em;  
}  
  
#form-container input{  
 box-sizing: border-box;  
 text-align: left;  
 padding: 10px;  
 border-radius: 15px;  
 color: var(--input-text-color);  
 margin: 5px;  
 font-size: 1.1em;  
 background-color: var(--inputs-bg-color);  
}  
  
#form-container a{  
 color: var(--form-text-color);  
}  
  
#form-container form{  
 box-sizing: border-box;  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding: 0;  
}  
  
#submit-button{  
 width: 100%;  
 padding: 10px;  
 border-radius: 15px;  
 background-color: var(--submit-button-color);  
 color: var(--form-text-color);  
 border-style: solid;  
 border-color: var(--form-text-color);  
 text-align: center;  
 font-size: 1.1em;  
 transition: background-color 0.5s, border-color 0.7s, font-size 1s;  
 cursor: pointer;  
}  
  
#form-container form #submit-button:*hover*{  
 background-color: var(--submit-button-hover-color);  
 border-style: solid;  
 border-color: var(--form-bg-color);  
 font-size: 1.2em;  
}

profile-styles.css:

:*root* {  
 --form-bg-color: #043c75;  
 --form-text-color: #f1f1f1;  
 --input-text-color: #333;  
 --input-text-incorrect-color: #f33;  
 --submit-button-color: #0968a2;  
 --submit-button-hover-color: #2080b0;  
 --submit-button-incorrect-color: #f33;  
 --submit-button-disabled-color: #555;  
 --inputs-incorrect-bg-color: #ffb0d0;  
 --inputs-bg-color: #f1f1f1;  
 --usercard-bg: #dfe4ff;  
 --usercard-hover-bg: #e8e8ff;  
 --onhover-scale-rate: 1.1;  
 --transition-time: .4s;  
}  
  
html, body{  
 min-height: 100%;  
 min-width: 390px;  
 display: block;  
 box-sizing: border-box;  
}  
  
#profiles-container{  
 width: 100%;  
 display: flex;  
 flex-wrap: wrap;  
 gap: 20px;  
 justify-content: center;  
 padding: 20px;  
 box-sizing: border-box;  
}  
  
#profile-container{  
 width: 100%;  
 box-sizing: border-box;  
 background-color: var(--form-bg-color);  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding: 1%;  
 */\*border-radius: 15px;\*/* color: var(--form-text-color);  
 font-family: Arial, Helvetica, sans-serif;  
 font-size: 1.2em;  
}  
  
#profile-container h1{  
 box-sizing: border-box;  
 text-align: center;  
 margin: 15px;  
 font-size: 2em;  
}  
  
#profile-container a{  
 color: var(--form-text-color);  
}  
  
#profile-container form{  
 box-sizing: border-box;  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding: 0;  
 width: 100%;  
}  
  
#submit-button{  
 width: 100%;  
 padding: 10px;  
 border-radius: 15px;  
 margin: 5px;  
 box-sizing: border-box;  
 background-color: var(--submit-button-color);  
 color: var(--form-text-color);  
 border-style: solid;  
 border-color: var(--form-text-color);  
 text-align: center;  
 font-size: 1.1em;  
 transition: background-color 0.5s, border-color 0.7s, font-size 1s;  
 cursor: pointer;  
}  
  
#profile-container form #submit-button:*hover*{  
 background-color: var(--submit-button-hover-color);  
 border-style: solid;  
 border-color: var(--form-bg-color);  
 font-size: 1.2em;  
}  
  
.data-element-container{  
 width: 100%;  
 display: flex;  
 flex-direction: row;  
}  
  
.data-element-name{  
 display: flex;  
 flex: 2;  
 min-width: 90px;  
 text-align: center;  
 justify-content: center;  
 align-items: center;  
}  
  
#profile-container input{  
 box-sizing: border-box;  
 flex: 15;  
 text-align: left;  
 padding: 10px;  
 border-radius: 15px;  
 color: var(--input-text-color);  
 margin: 5px;  
 font-size: 1.1em;  
 background-color: var(--inputs-bg-color);  
}  
  
.userCard {  
 background-color: var(--usercard-bg);  
 padding: 20px 50px;  
 border: 1px solid;  
 border-radius: 8px;  
 text-align: center;  
 transition: transform var(--transition-time), background-color var(--transition-time);  
}  
  
.userCard:*hover* {  
 transform: scale(var(--onhover-scale-rate));  
 background-color: var(--usercard-hover-bg);  
}

validations.js:

function \_validateUsername(usrname)  
{  
 return (usrname.value != null && (/^\w{3,20}$/.test(username.value)));  
}  
  
function \_validatePasswd(passw)  
{  
 return passw.value != null && passw.value.length > 5 && passw.value.match(/[A-Z]/)?.length && passw.value.match(/[a-z]/)?.length && passw.value.match(/\d/)?.length;  
}  
  
function \_validateName(fullname)  
{  
 if (fullname.value != null && (/^[A-ZА-ЯІҐЄЇ][a-zа-яіґєї]+ [A-ZА-ЯІҐЄЇ]\.[A-ZА-ЯІҐЄЇ]\.$/.test(fullname.value)))  
 {  
 fullname.style.background = inp\_bg\_color;  
 return true;  
 }  
  
 fullname.style.background = incorrect\_inp\_bg\_color;  
 return false;  
}  
  
function \_validateIdCard(IdCard)  
{  
 if (IdCard.value != null && (/^\d{9}$/.test(IdCard.value)))  
 {  
 IdCard.style.background = inp\_bg\_color;  
 return true;  
 }  
  
 IdCard.style.background = incorrect\_inp\_bg\_color;  
 return false;  
}  
  
function \_validateFaculty(faculty)  
{  
 if (faculty.value != null && (/^[A-ZА-ЯІҐЄЇ]{3,5}$/.test(faculty.value)))  
 {  
 faculty.style.background = inp\_bg\_color;  
 return true;  
 }  
  
 faculty.style.background = incorrect\_inp\_bg\_color;  
 return false;  
}  
  
function \_validateBirthDate(birthdate)  
{  
 if (birthdate.value != null && Date.parse(birthdate.value) != null &&  
 birthdate.value.length === 10 && Date.parse(birthdate.value) < Date.now())  
 {  
 birthdate.style.background = inp\_bg\_color;  
 return true;  
 }  
  
 birthdate.style.background = incorrect\_inp\_bg\_color;  
 return false;  
}  
  
function \_validateAddress(address)  
{  
 if (address.value != null && (/^м. [А-ЯІҐЄЇ][а-яіґєї]+([- ][А-ЯІҐЄЇ][а-яіґєї]+)?$/.test(address.value)))  
 {  
 address.style.background = inp\_bg\_color;  
 return true;  
 }  
  
 address.style.background = incorrect\_inp\_bg\_color;  
 return false;  
}

sign-up.js:

const rootStyles = getComputedStyle(document.querySelector(':root'));  
let inp\_bg\_color = rootStyles.getPropertyValue('--inputs-bg-color');  
let incorrect\_inp\_bg\_color = rootStyles.getPropertyValue('--inputs-incorrect-bg-color');  
let submit\_bg = rootStyles.getPropertyValue('--submit-button-color');  
let submit\_dis\_bg = rootStyles.getPropertyValue('--submit-button-disabled-color');  
  
  
const submit = document.getElementById('submit-button');  
const form1 = document.getElementById('form1');  
const form2 = document.getElementById('form2');  
  
const username = document.getElementById('username');  
const passwd = document.getElementById('passwd');  
const full\_name = document.getElementById('full\_name');  
const id\_card = document.getElementById('id\_card');  
const faculty = document.getElementById('faculty');  
const birthdate = document.getElementById('birthdate');  
const address = document.getElementById('address');  
  
const fields = [full\_name, id\_card, faculty, birthdate, address]  
const fildIsValid = [false, false, false, false, false];  
const validationFuncs = [\_validateName, \_validateIdCard, \_validateFaculty, \_validateBirthDate, \_validateAddress]  
  
  
submit.addEventListener('click', GoNext);  
  
function GoNext(){  
 if (username.value && passwd.value){  
 if (!\_validateUsername(username)){  
 alert("Incorrect username format! Please, use only latin letters, digits and underlines!")  
 }  
 else if (!\_validatePasswd(passwd)){  
 alert("Insecure password! Please, use digits, uppercase and lowercase latin letters!")  
 }  
 else{  
 fetch(`/api/sign-up/${username.value}`).then((resp) => resp.json()).then(  
 data => {  
 if (data.available){  
 form1.style.display = 'none';  
 form2.style.display = 'flex';  
 submit.innerText = 'Sign Up';  
 submit.removeEventListener('click', GoNext);  
 for (let i = 0; i < fields.length; i++){  
 fields[i].addEventListener('input', () => OnChanged(i));  
 }  
 submit.addEventListener('click', TrySignUp);  
 }  
 else{  
 alert('username taken');  
 }  
 }  
 );  
 }  
 }  
}  
  
function OnChanged(i){  
 fildIsValid[i] = validationFuncs[i](fields[i]);  
 if (fildIsValid.every((el) => el)) {  
 submit.disabled = false;  
 submit.style.background = submit\_bg;  
 }  
 else {  
 submit.disabled = true;  
 submit.style.background = submit\_dis\_bg;  
 }  
}  
  
function TrySignUp(){  
 axios.post("/api/sign-up", {  
 login: username.value,  
 passwd: passwd.value,  
 full\_name: full\_name.value,  
 id\_card: id\_card.value,  
 faculty: faculty.value,  
 birthdate: birthdate.value,  
 address: address.value  
 }, {  
 headers: {  
 "Content-Type": "multipart/form-data",  
 },  
 })  
 .then((response) => response.data)  
 .then((data) => {  
 if (data.status === "Success"){  
 window.location.replace("/");  
 }  
 else{  
 alert("Something went wrong")  
 }  
 });  
}  
  
function PrintData(data){  
 const classname = "data-out-container";  
 if (document.body.getElementsByClassName(classname).length > 2){  
 document.body.getElementsByClassName(classname)[0].remove();  
 }  
 let dataOutput = "<div class="+classname+"><h1>Введені дані:</h1>" +  
 "<p><b>ПІБ:</b> " + data.fullname.value + "</p>" +  
 "<p><b>ID-карта:</b> " + data.IdCard.value + "</p>" +  
 "<p><b>Факультет:</b> " + data.faculty.value + "</p>" +  
 "<p><b>Дата народження:</b> " + data.birthdate.value + "</p>" +  
 "<p><b>Адреса:</b> " + data.address.value + "</p></div>";  
 document.body.innerHTML += dataOutput;  
}

log-in.js:

const username = document.getElementById('username');  
const passwd = document.getElementById('passwd');  
const submit = document.getElementById('submit-button');  
  
const rootStyles = getComputedStyle(document.querySelector(':root'));  
let incorrect\_button\_color = rootStyles.getPropertyValue('--submit-button-incorrect-color');  
let correct\_button\_color = rootStyles.getPropertyValue('--submit-button-color');  
  
  
submit.addEventListener('click', () => {TryLogIn();});  
  
function TryLogIn(){  
 axios.post("/api/log-in", {  
 username: username.value,  
 passwd: passwd.value  
 }, {  
 headers: {  
 "Content-Type": "multipart/form-data",  
 },  
 })  
 .then((response) => response.data)  
 .then((data) => {  
 if (data.status === "Success"){  
 window.location.replace("/");  
 }  
 else{  
 submit.innerText = "Incorrect"  
 submit.style.backgroundColor = incorrect\_button\_color;  
 submit.removeEventListener('click', () => {TryLogIn();});  
 username.addEventListener('input', () => discard\_incorrect\_state());  
 passwd.addEventListener('input', () => discard\_incorrect\_state());  
 }  
 });  
}  
  
function discard\_incorrect\_state(){  
 submit.innerText = "Log in"  
 submit.style.backgroundColor = correct\_button\_color;  
 submit.addEventListener('click', () => {TryLogIn();});  
 username.removeEventListener('change', () => discard\_incorrect\_state());  
 passwd.removeEventListener('change', () => discard\_incorrect\_state());  
}

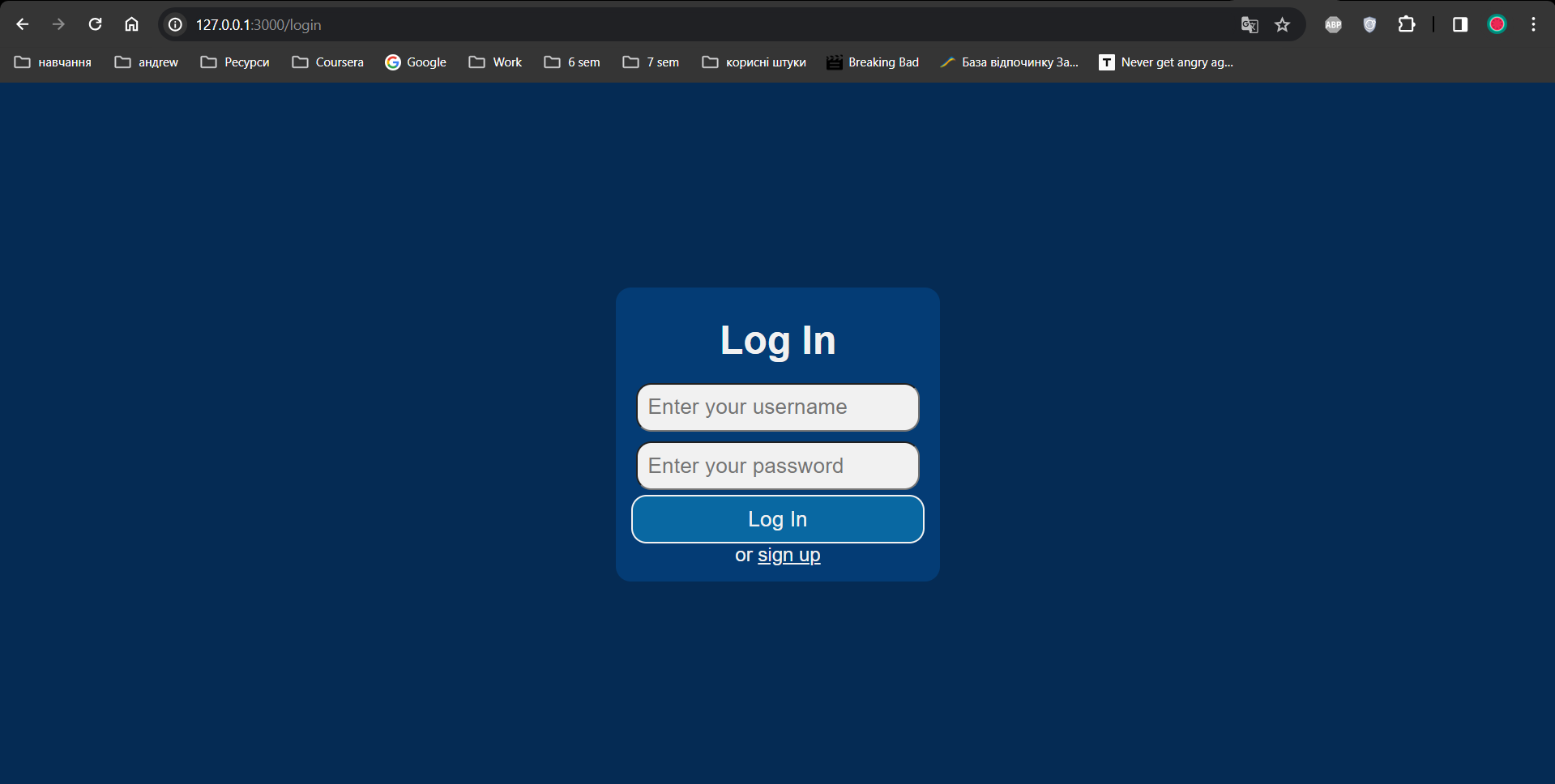
**3. Отримані результати**

Запустимо сервер командою:

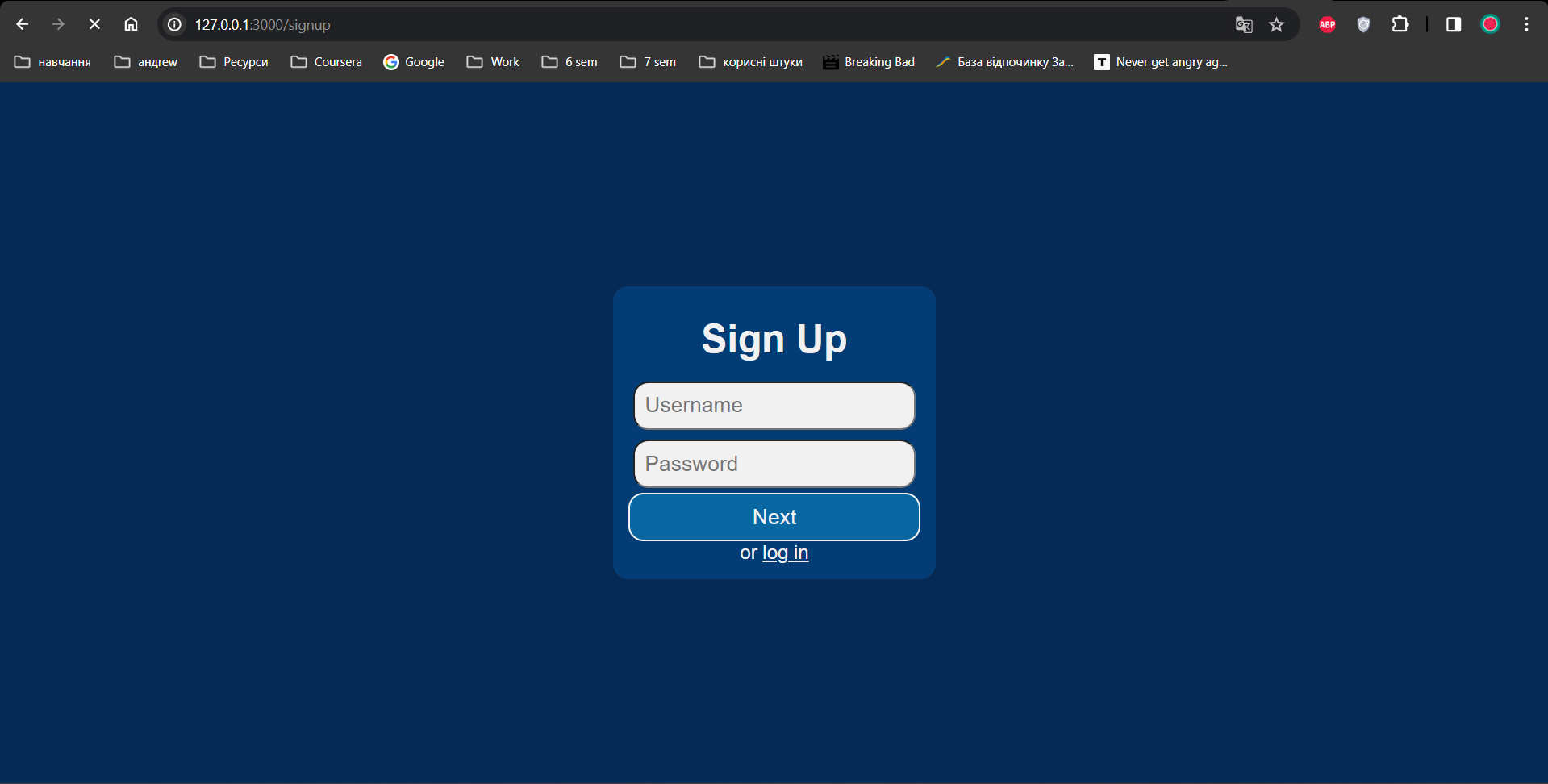
app.run("0.0.0.0", port=3000, debug=False)

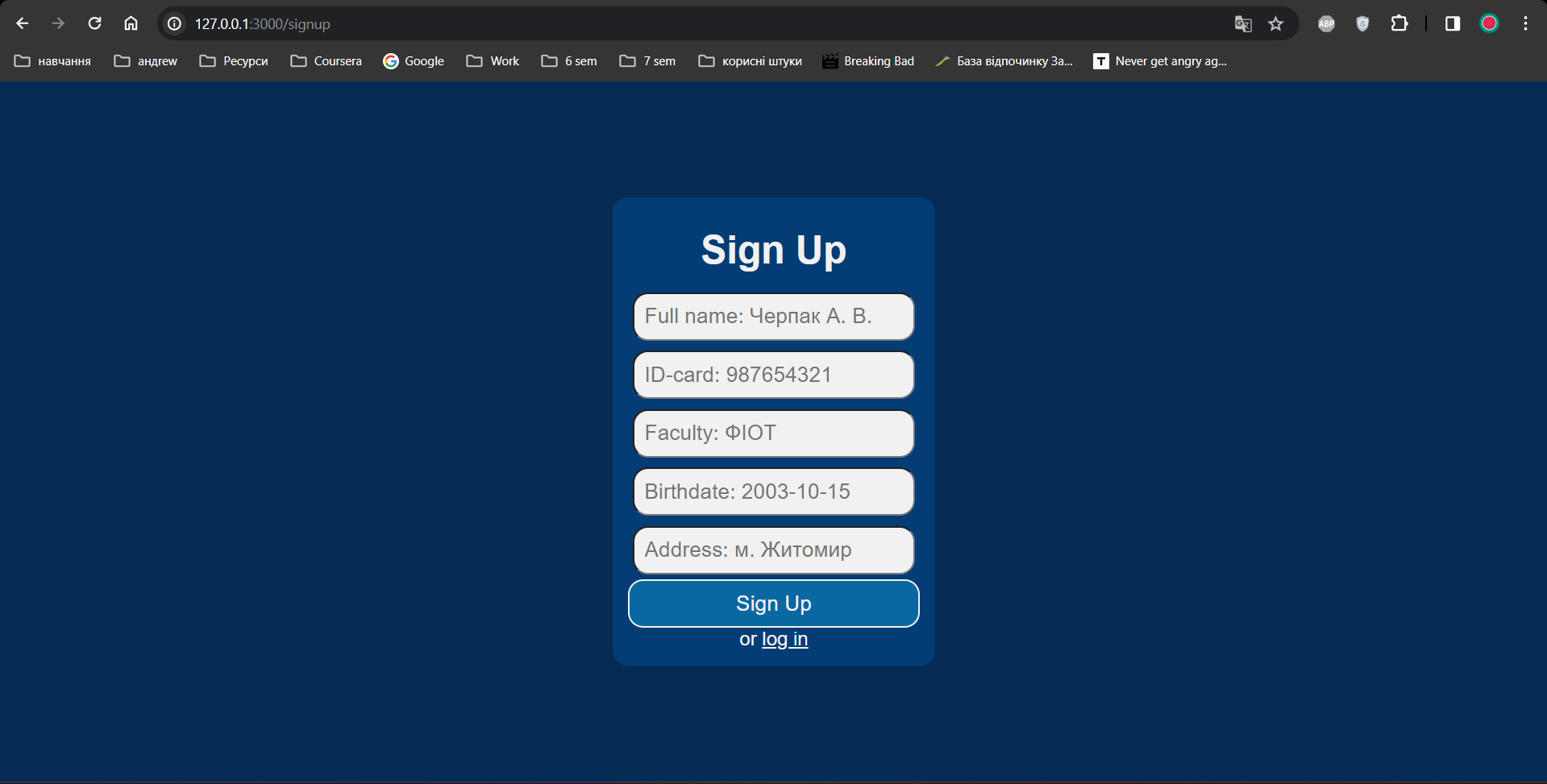
Перейдемо за посиланням: http://127.0.0.1:3000/

Нас переадресує на таку сторінку:

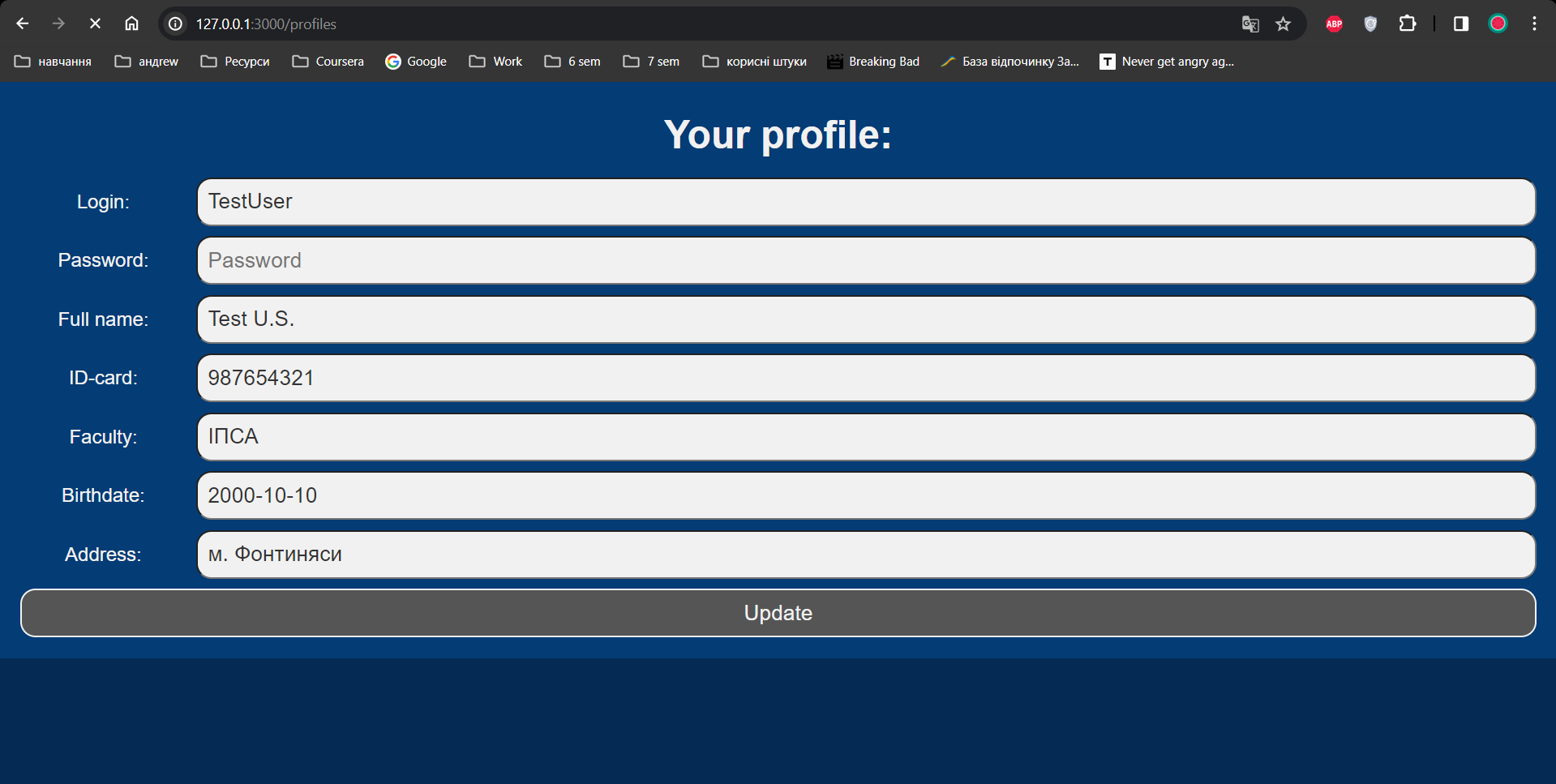


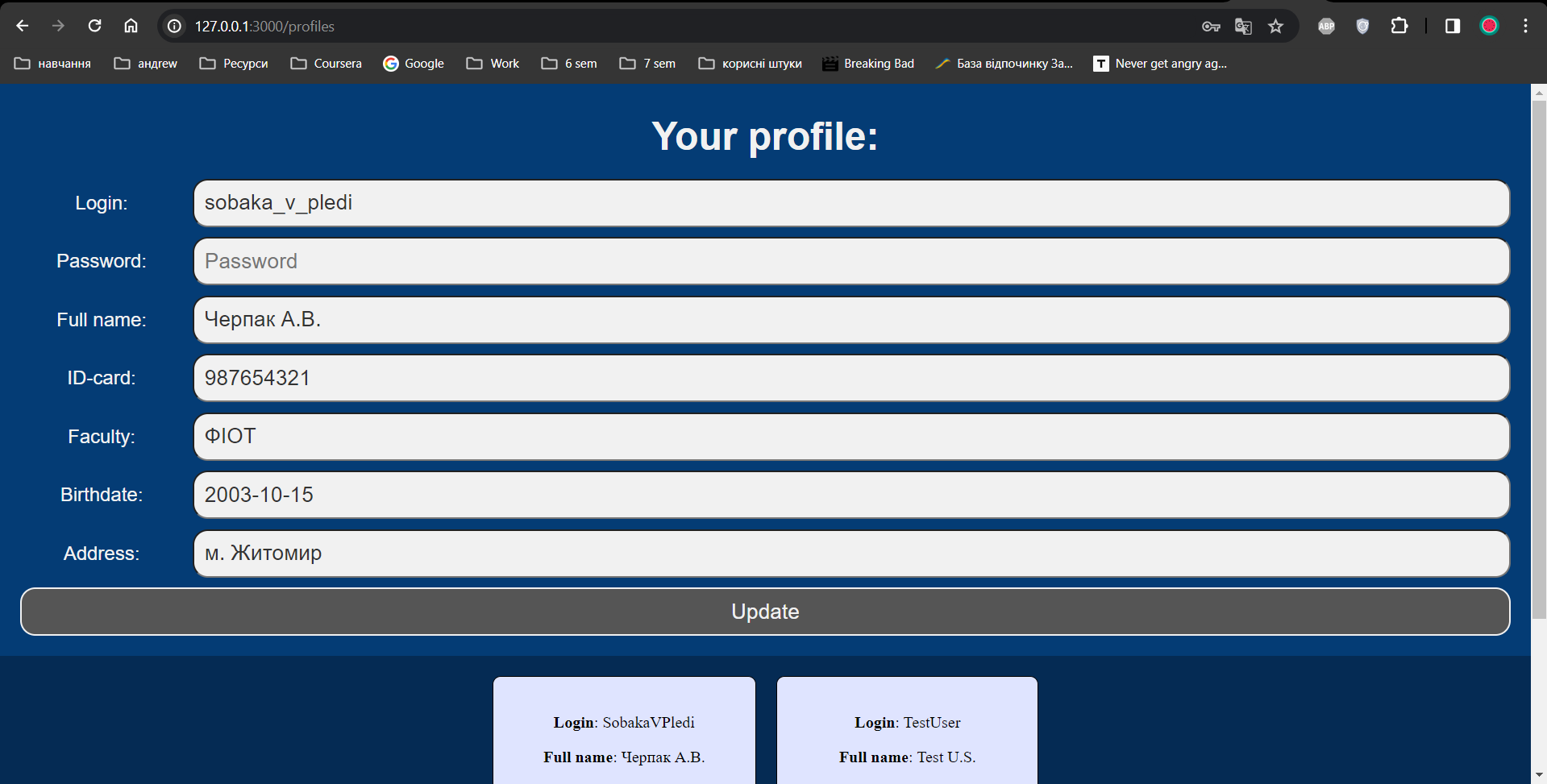
Якщо акаунт не було ще створено – можемо натиснути на sign up та потрапити на іншу сторінку:

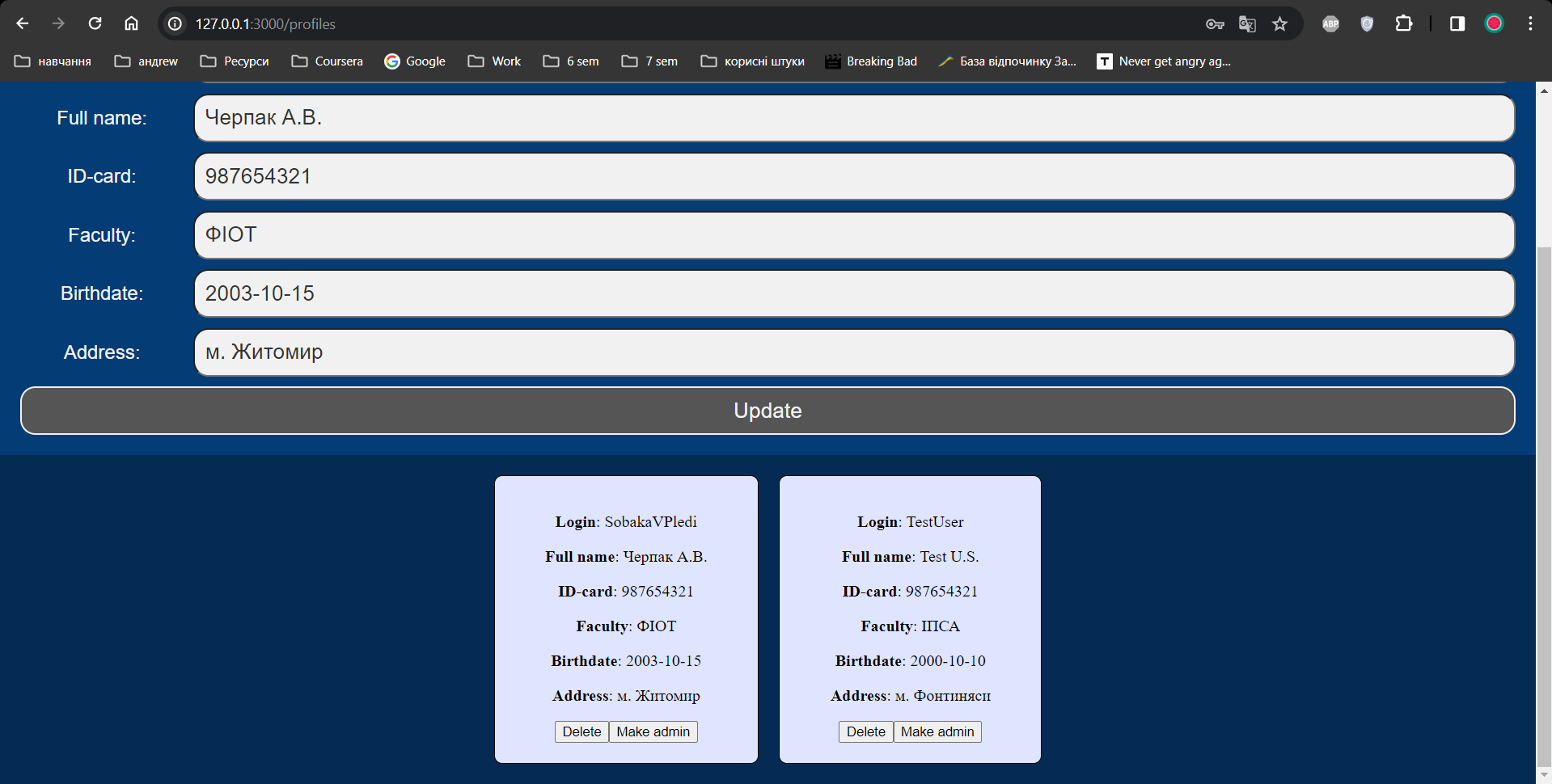




Після введення усіх необхідних даних потрапляємо на сторінку профілів, де можемо редагувати свої дані, а якщо ми адміни – то ще й переглядати дані інших користувачів, редагувати або видаляти їх:



Для адмінів:



На цій сторінці ми можемо видалити користувачів або призначити їх адміністраторами.

Посилання на репозиторій gitgub:

<https://github.com/CherpakAndrii/web-basics/tree/master/Lab5>

**Висновок**

Під час виконання даної лабораторної роботи я навчився створювати веб-застосунки з використанням фреймворку Flask.py, взаємодіяти з базами даних та працювати з аутентифікацією та авторизацією користувачів. Також було удосконалено навики верстки, роботи з промісами та бібліотекою axios.js.