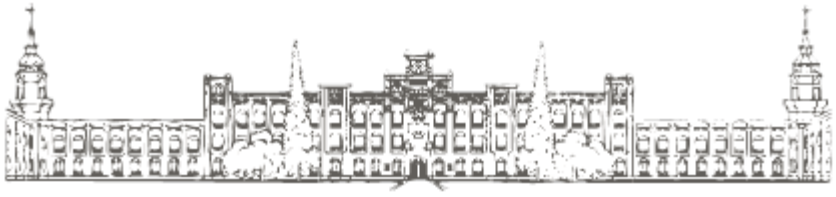


Міністерство освіти і науки України

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:

Доц. Голубєв Л. П.

Виконав:

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
A. B.">
    <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;
}
```

```css
#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:

```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-ZA-ЯIЇЄЇ][a-za-яiїєї]+ [A-ZA-ЯIЇЄЇ]\.[A-ZA-
ЯIЇЄЇ]\.$/.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-ZA-ЯIЇЄЇ]{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-ЯІЇЄЇ][а-яіїєї]+([- ][А-ЯІЇЄЇ][а-яіїєї]+)?$/.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());
}
```

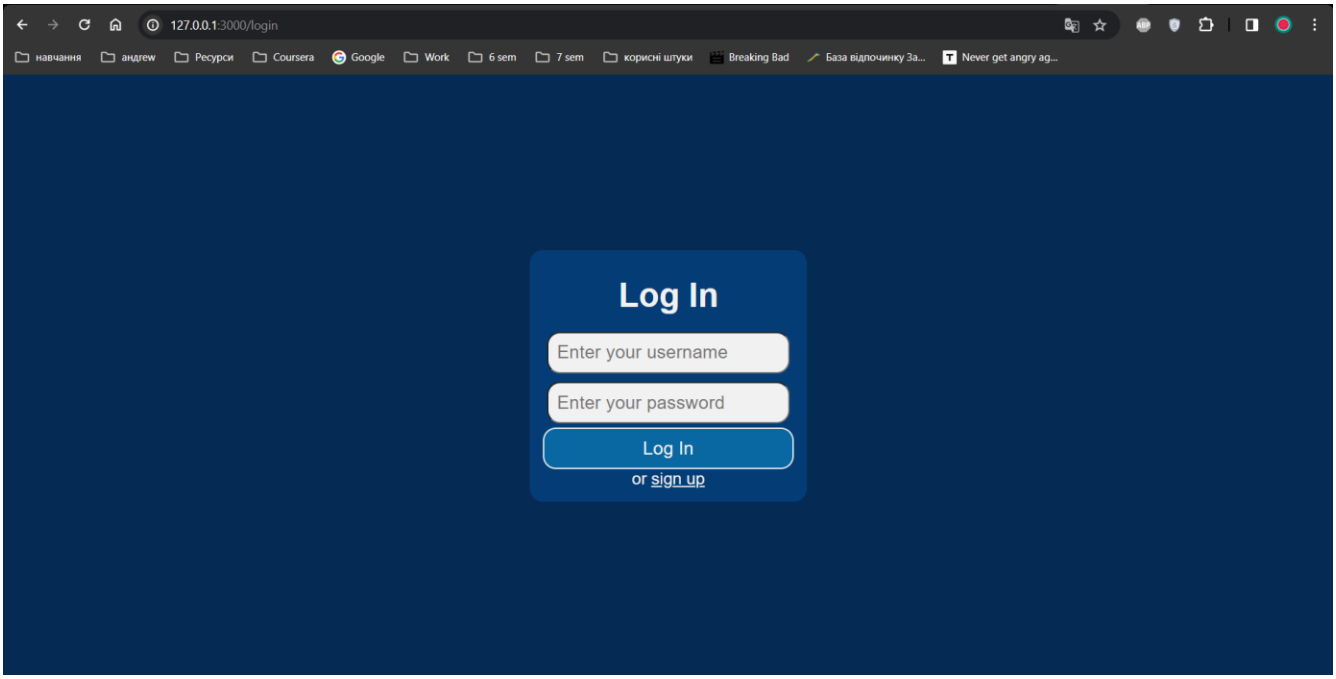
3. Отримані результати

Запустимо сервер командою:

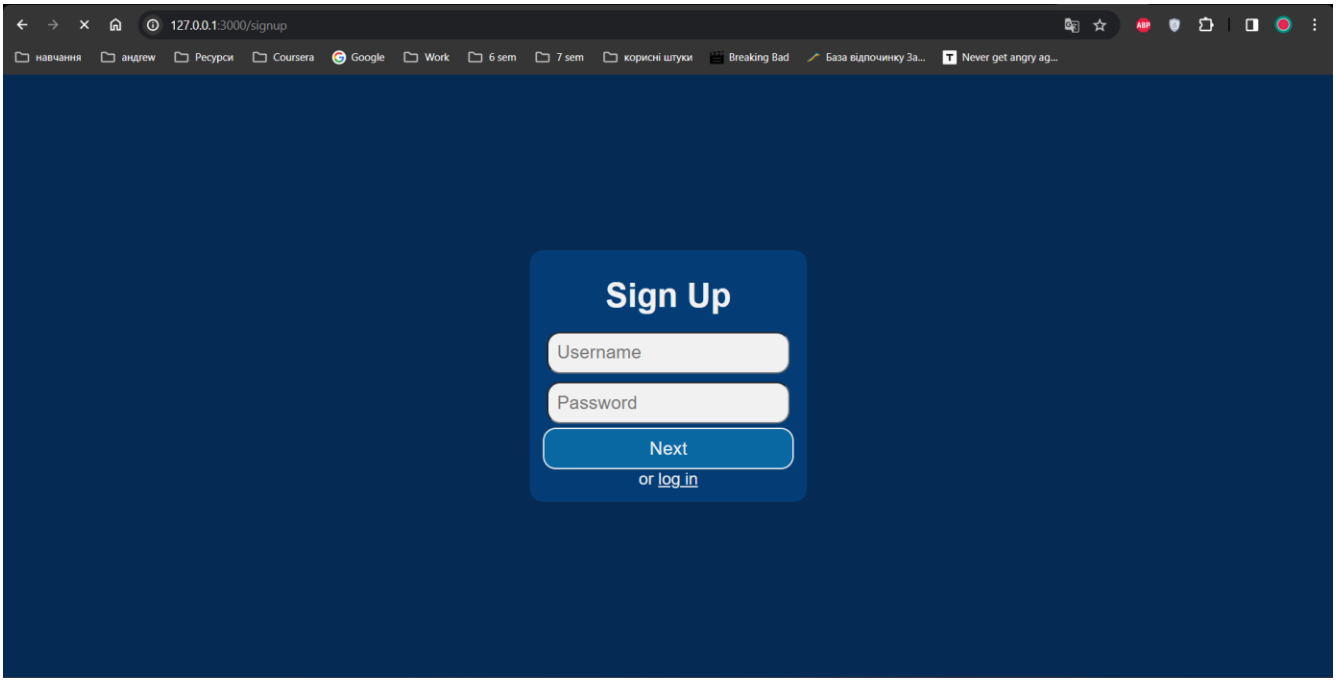
```
app.run("0.0.0.0", port=3000, debug=False)
```

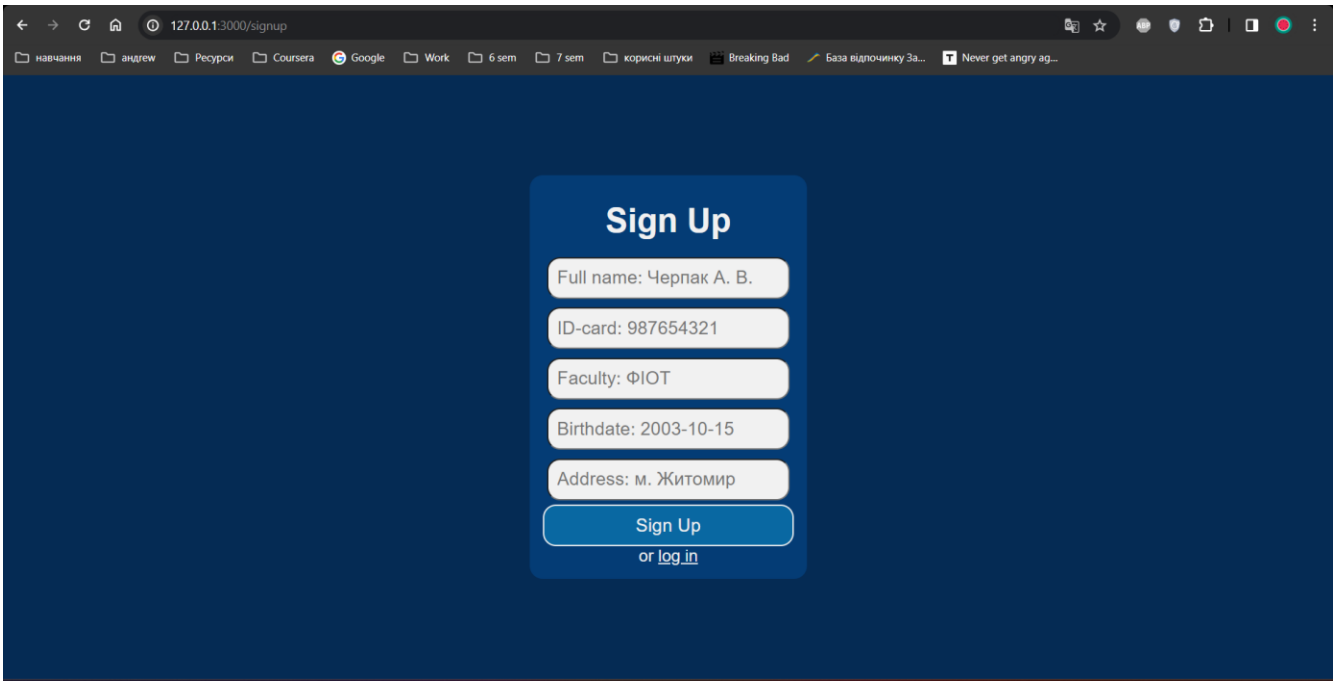
Перейдемо за посиланням: http://127.0.0.1:3000/

Нас переадресує на таку сторінку:

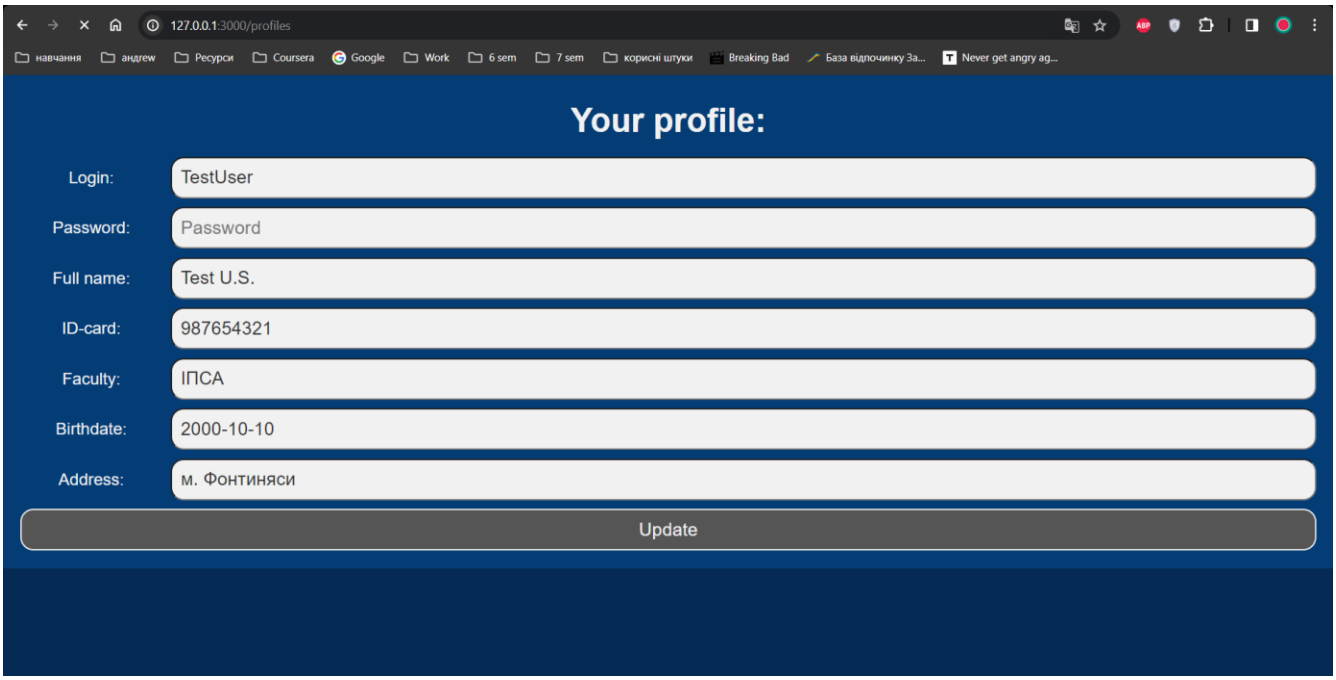


Якщо акаунт не було ще створено – можемо натиснути на sign up та потрапити на іншу сторінку:

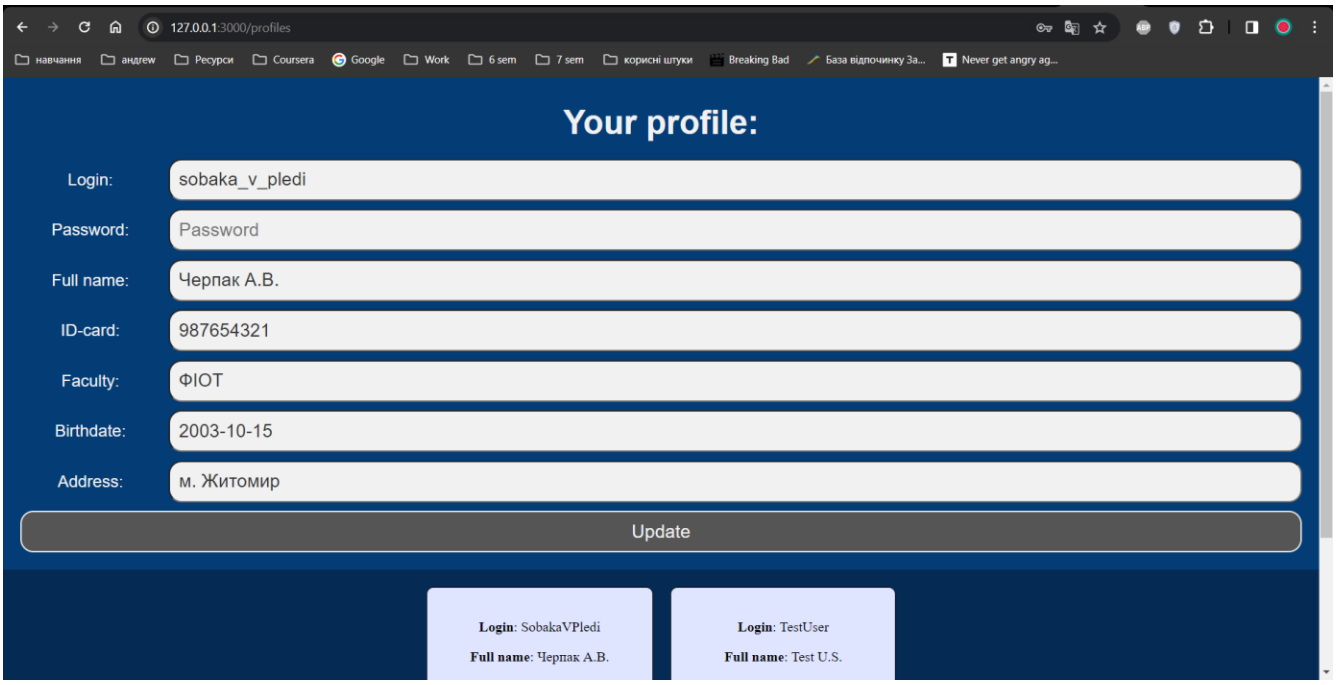


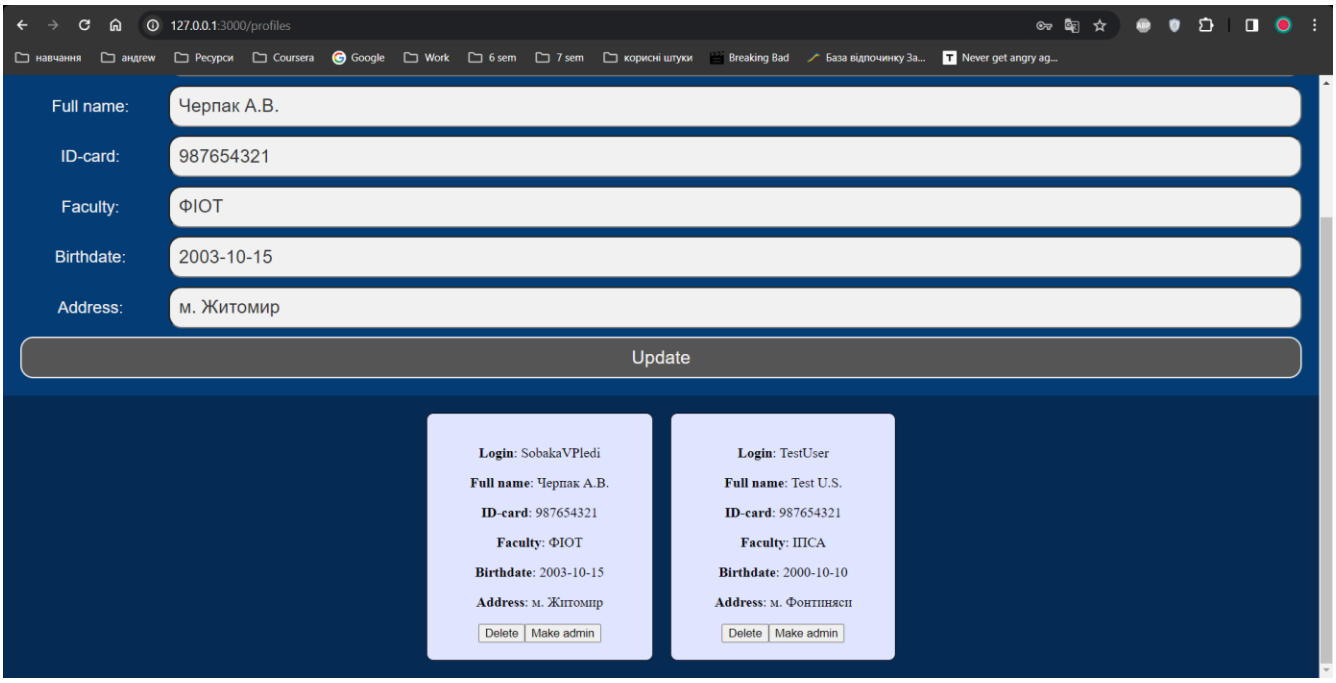


Після введення усіх необхідних даних потрапляємо на сторінку профілів, де можемо редагувати свої дані, а якщо ми адміни – то ще й переглядати дані інших користувачів, редагувати або видаляти їх:



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

20

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.