Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З комп’ютерного практикуму № 4 з дисципліни

**«Безпека програмного забезпечення»**

Виконав студент ІП-13 Музичук Віталій Андрійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Соколовський Владислав Володимирович

(прізвище, ім'я, по батькові)

Дата оцінювання \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2024

**Лабораторна робота 4**

**Тема:** Засвоювання базових навичок OAuth2 авторизаційного протокола.

**Завдання:** Використовуючи наведені налаштування з лабораторної роботи 2-3 та приведених запитів модифікувати аплікейшен <https://github.com/Kreolwolf1/auth_examples/tree/main/token_auth>. Використовую-чи перевірку юзера та отримання токена з auth0 (password grant type) Надати код модифікованного аплікейшена.

**Для отримання додаткового балу:** додатково розшири аплікайшен створенням юзера та перевіркою життя токена (у разі близького завершення – оновити токен використовуючи refresh-token grant type).

**Хід роботи**

Код даного застосунку можна переглянути на гітхабі за посиланням

<https://github.com/muzy4ukV/softwareSecurityLabsKPI/tree/main/%D0%9B%D0%B0%D0%B1%D0%B0%204>

Змінимо виконання запиту для логінації користувача так, щоб додаток на основі username та password отримував user\_access\_token.

Код буде виглядати наступним чином

app.post('/api/login', (req, res) => {

    const { login, password } = req.body;

    let options = {

        uri:  `https://${domain}/oauth/token`,

        body: JSON.stringify(details),

        method: 'POST',

        headers: {

                    'Content-Type': 'application/json'

                }

    };

    request(options, function (error, response, body) {

        if(error){

            res.status(401).send();

        }

        if (!error && response.statusCode == 200) {

            const parsedBody = JSON.parse(body);

            let user\_details = {

                grant\_type: 'http://auth0.com/oauth/grant-type/password-realm',

                username: login,

                password: password,

                audience:  `https://${domain}/userinfo`,

                client\_id: details.client\_id,

                client\_secret: details.client\_secret,

                realm: 'Username-Password-Authentication',

                scope: 'offline\_access'

            };

            let user\_options = {

                uri: `https://${domain}/oauth/token`,

                body: JSON.stringify(user\_details),

                method: 'POST',

                headers: {

                            'Content-Type': 'application/json',

                            'Authorization' : 'Bearer ' + parsedBody.access\_token

                        }

            };

            request(user\_options, function (error, response, body) {

                if(error){

                    res.status(401).send();

                }

                if(response.statusCode == 403){

                    return res.status(403).send(body);

                }

                if (!error && response.statusCode == 200) {

                    var body\_res = JSON.parse(body);

                    res.json({ token: body\_res.access\_token,

                               refresh\_token: body\_res.refresh\_token });

                }

            });

        }

    });

});

Тобто спочатку ми робимо запит на адресу https://${domain}/oauth/token, щоб отримати client\_access\_token, який дозволить додатку перевіряти логін й пароль користувача. Наступним кроком буде запит, на отримання user\_access\_token, який надалі можемо використовувати щоб отримати доступ до API /userinfo й переглядати інформацію про користувача.

Тепер звичайна перевірка на запиті GET виконується за допомогою отриманого user\_access\_token

app.get('/', (req, res) => {

    let auth = req.get("Authorization");

    if(!auth){

        res.sendFile(path.join(\_\_dirname+'/index.html'));;

        return;

    }

    const options = {

        method: "GET",

        uri: `https://${domain}/userinfo`,

        headers: { "Authorization": "Bearer " + auth }

      };

      request(options, function (error, response, body) {

              if (!error && response.statusCode == 200) {

                  var body\_res = JSON.parse(body);

                  return res.json({

                      username: body\_res.nickname,

                      logout: `http://localhost:${port}/logout`

                  });

              }

              else{

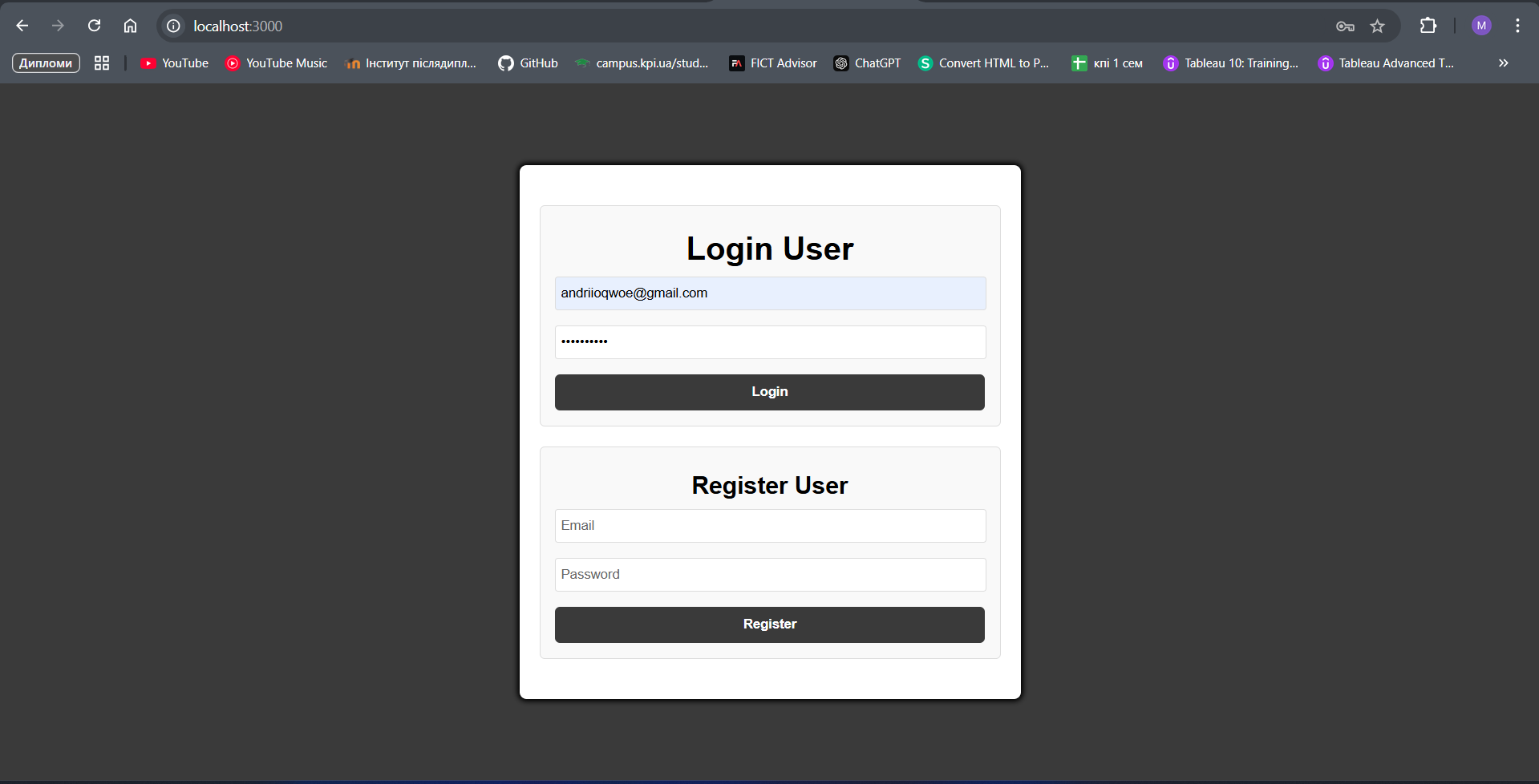
                  res.status(401).send(body);

              }

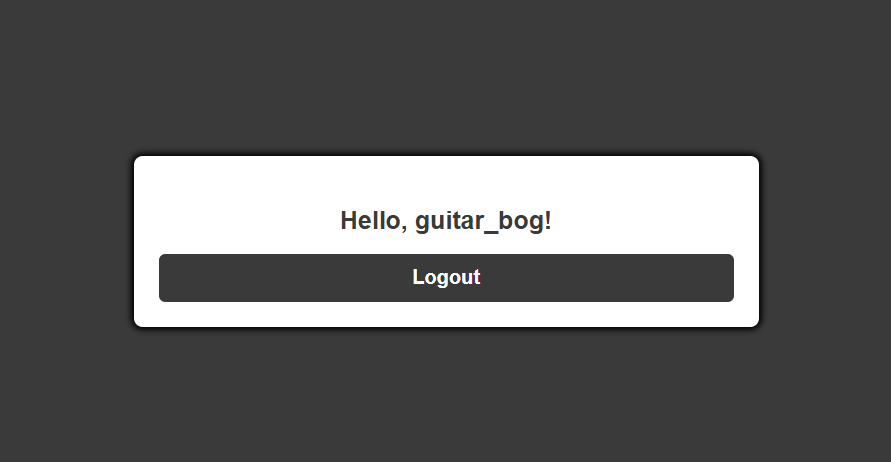
      });

})

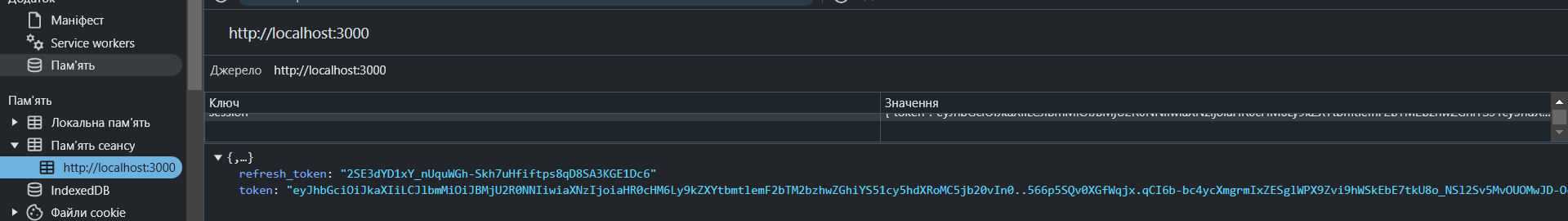
Спробуємо залогінитися як користувач, що був створений у попередній лабораторній.



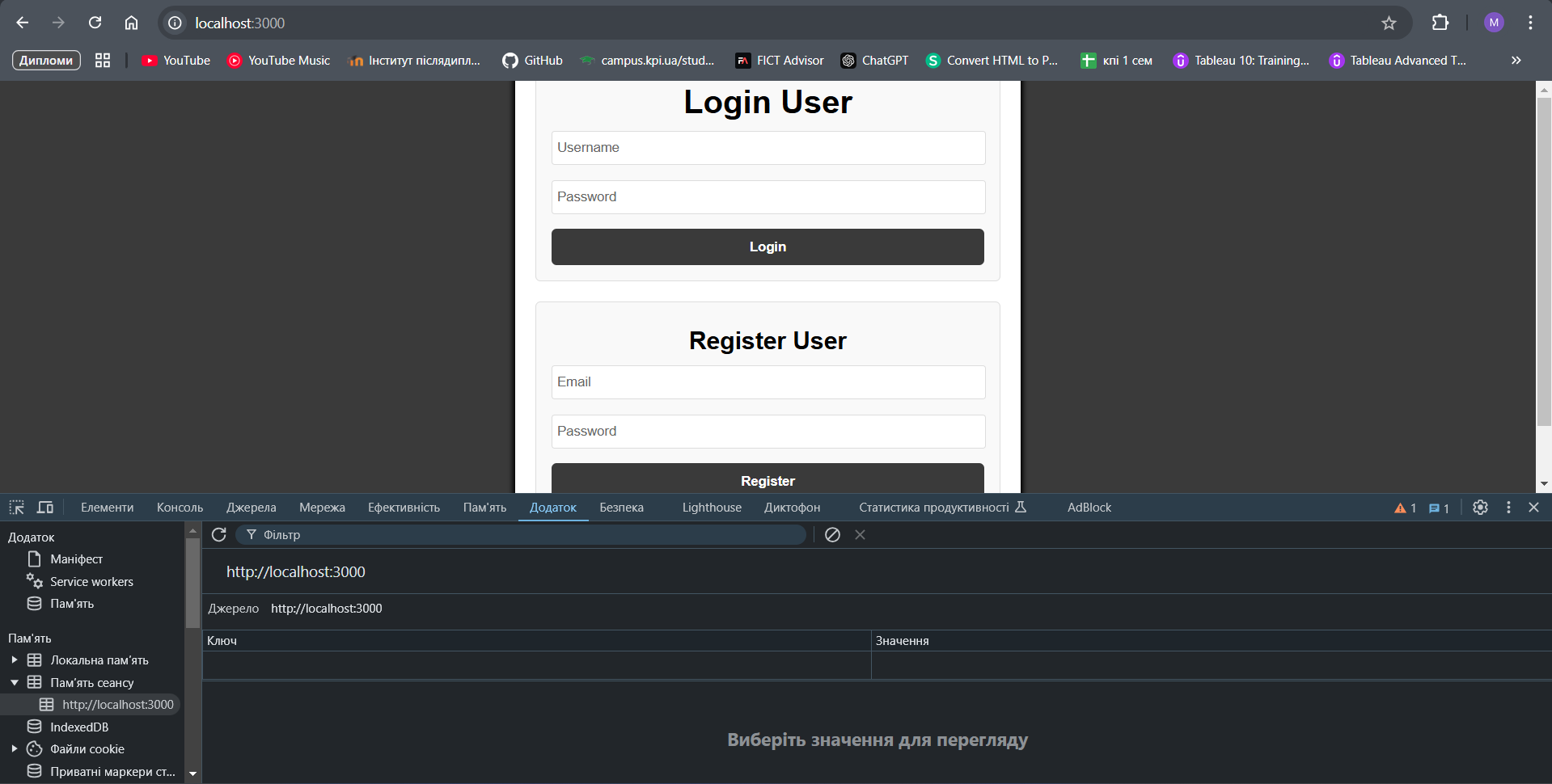
Вхід у систему пройшов успішно.



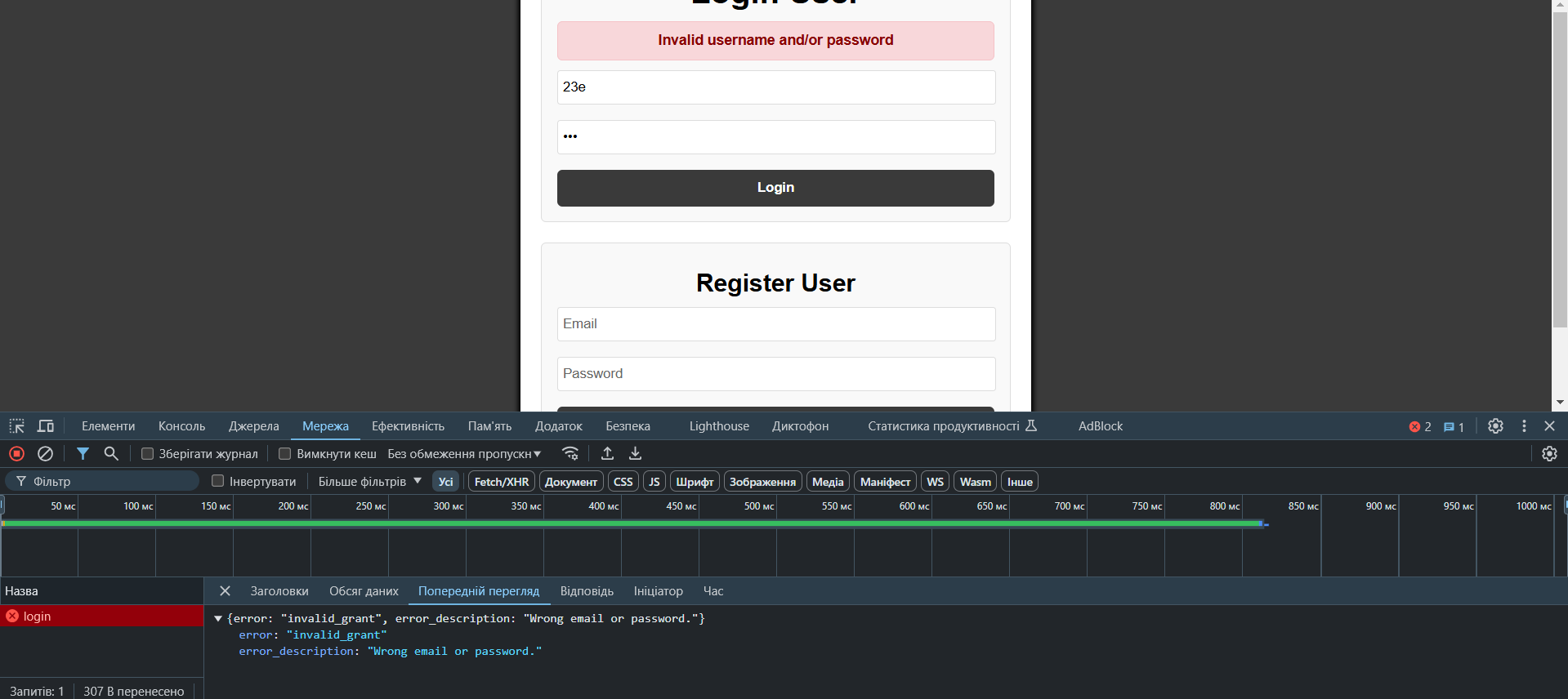
Можемо переглянути access\_token, який зберігається у сесії користувача.



При виході з системи – sessionStorage очищається й ми повретаємося на головну сторінку.



Спробуємо ввести невірні дані для входу користувача



Отримали 403 помилку й повідомлення про те, що credentials user is wrong.

**Додаткове завдання**

Додамо функціонал, що буде створювати користувача. Для цього попередньо вже була додана ще одна форма, яка надсилатиме запит POST на адресу api/register. Серврена частина запиту виглядає наступним чином

app.post('/api/register', (req, res) => {

    const { email, password } = req.body;

    if (!email || !password) {

        return res.status(400).json({ error: 'Missing required fields' });

    }

    const managementOptions = {

        method: 'POST',

        uri: `https://${domain}/oauth/token`,

        headers: { 'Content-Type': 'application/json' },

        body: JSON.stringify({

            client\_id: details.client\_id,

            client\_secret: details.client\_secret,

            audience: `https://${domain}/api/v2/`,

            grant\_type: 'client\_credentials',

        }),

    };

    request(managementOptions, (err, response, body) => {

        if (err || response.statusCode !== 200) {

            return res.status(500).json({ error: 'Failed to get Management API token' });

        }

        const accessToken = JSON.parse(body).access\_token;

        const createUserOptions = {

            method: 'POST',

            uri: `https://${domain}/api/v2/users`,

            headers: {

                'Authorization': `Bearer ${accessToken}`,

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({

                email,

                password,

                connection: 'Username-Password-Authentication',

            }),

        };

        request(createUserOptions, (err, response, body) => {

            if (err || response.statusCode >= 400) {

                return res.status(400).json({ error: 'Failed to register user', details: body });

            }

            res.status(201).json({ message: 'User registered successfully!' });

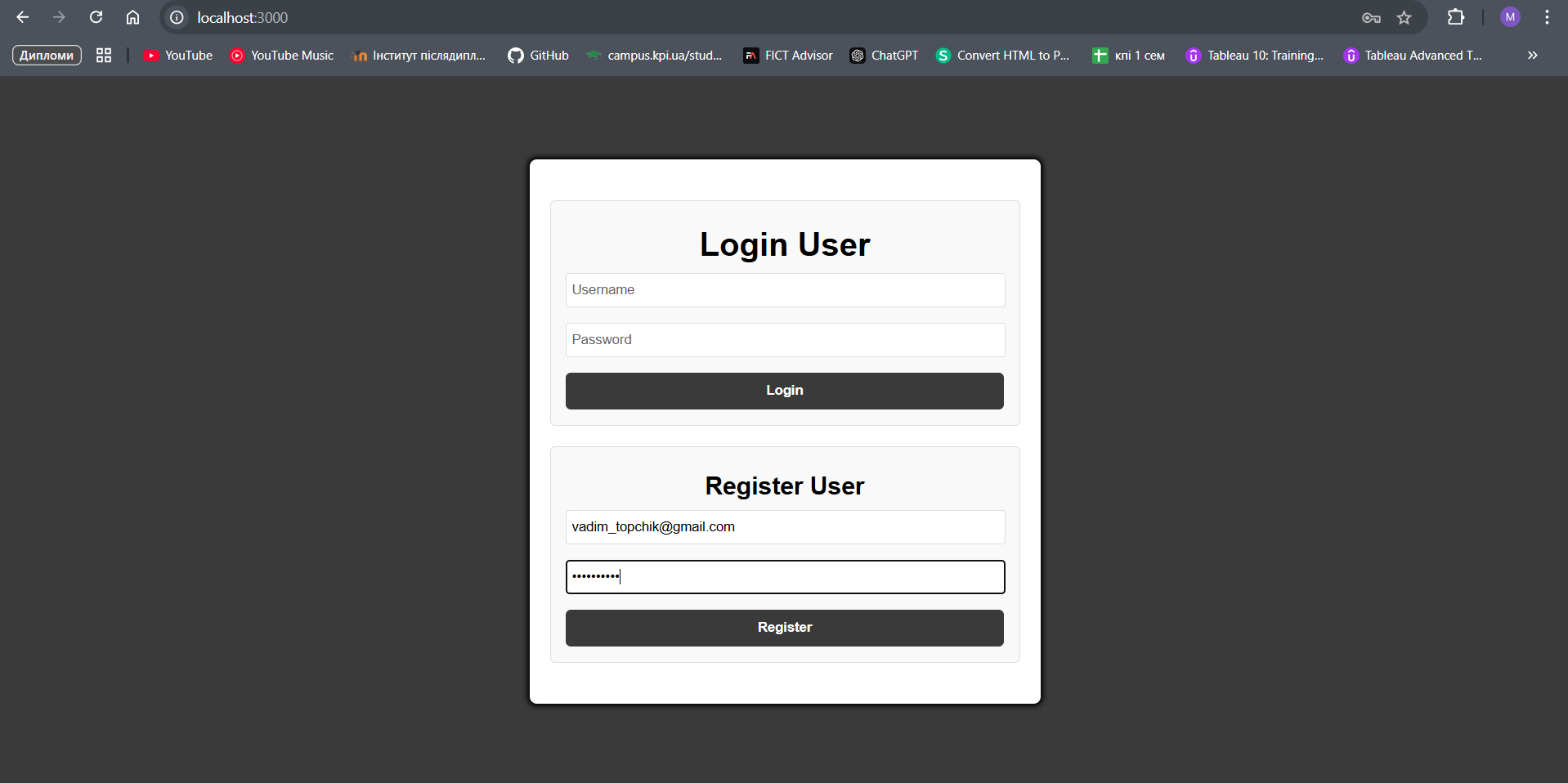
        });

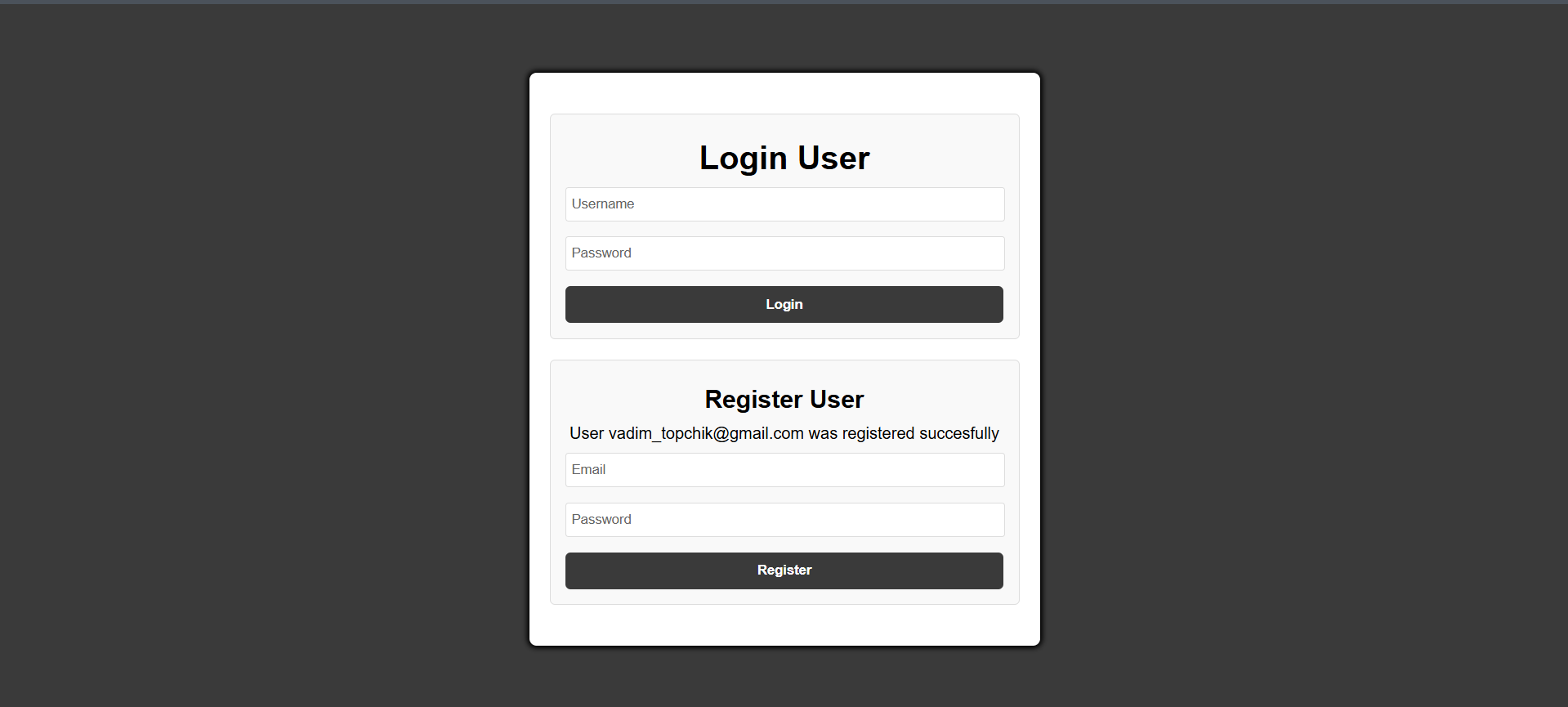
    });

});

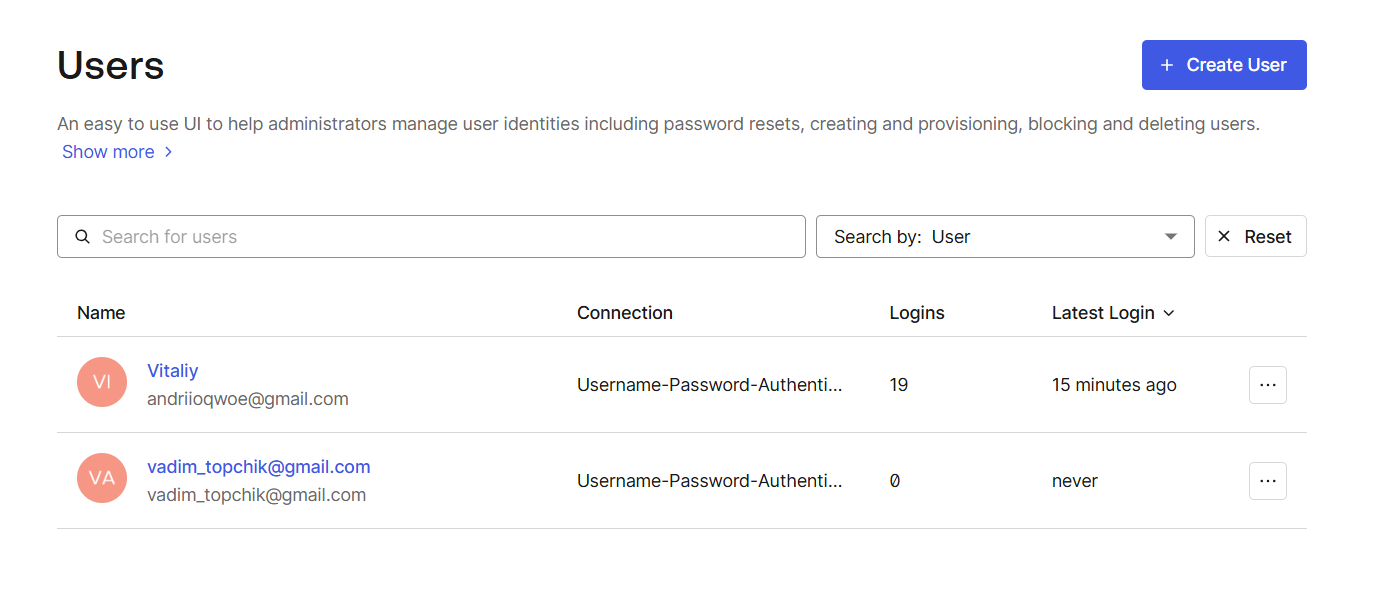
Отже як ми бачимо логіка схожа, спочатку отримаємо client\_access\_token, а потім за допомогою нього надсилаємо дані користувача на https://${domain}/api/v2/users і отримуємо підтвердження про успішну реєестрацію. Протестуємо даний функціонал

Реєструємо нового користувача

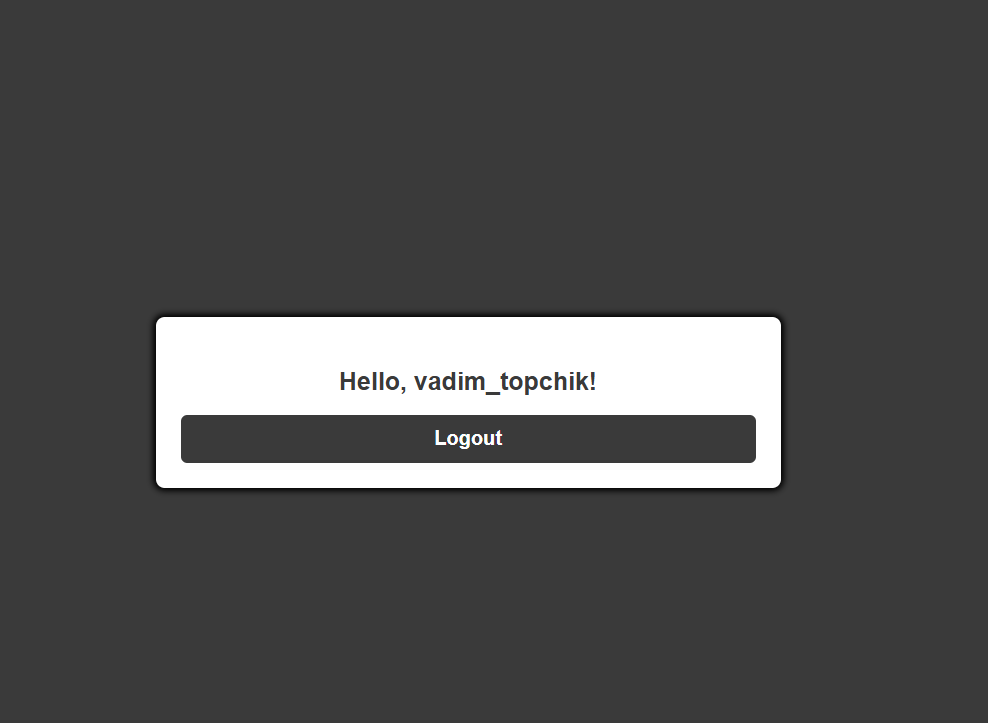




Реєстрація пройшла успішно. Можемо переконатися в цьому переглянувши користувачів додатку через сайт Auth0.



Логін для нового користувача проходить успішно



**Висновок**

У процесі виконання лабораторної роботи були засвоєні основи використання OAuth2 для авторизації та аутентифікації через Auth0. Було модифіковано аплікейшен для отримання `user\_access\_token` за допомогою password grant type та реалізовано перевірку користувача за цим токеном.

Також додано функціонал для реєстрації користувачів через API Auth0, що дозволяє створювати нових користувачів і виконувати логін за допомогою отриманих токенів. Всі етапи роботи супроводжувались обробкою помилок, зокрема для випадків невірних даних, коли повертається помилка 403.

Робота дозволила глибше ознайомитись з механізмами OAuth2 та налаштуванням безпечних систем авторизації.