МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

Про виконання лабораторної роботи №5

З дисципліни “Безпека програмного забезпечення”

На тему “ Засвоювання базових навичок роботи з валідацією токенів”

|  |  |
| --- | --- |
| Виконали:  Студенти групи ІП-15  Мєшков А. І. | Перевірила:  пос. Соколовський В. В. |

Київ 2024

ЛАБОРАТОРНА РОБОТА №5

Завдання:

Розширити **Лабораторну работу 4** перевіркой сігнатури JWT токена. Приклади SDK <https://auth0.com/docs/quickstart/backend>. У випадку асиметричного ключа, public є можливість отримати за посиланням <https://kpi.eu.auth0.com/pem>, aбо за формулою <https://[API_DOMAIN]/pem>

Надати код рішення.

## ХІД РОБОТИ

1. Було додано верифікацію JWT токена.

Index.html

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<title>Login</title>

<script *src*="https://unpkg.com/axios/dist/axios.min.js"></script>

</head>

<body>

<main *id*="main-holder">

<a *href*="/logout" *id*="logout">Logout</a>

<h1 *id*="login-header">Login</h1>

<div *id*="login-error-msg-holder">

<p *id*="login-error-msg">Invalid username <span *id*="error-msg-second-line">and/or password</span></p>

</div>

<form *id*="login-form" *action*="/api/login" *method*="post">

<input *type*="text" *name*="login" *id*="username-field" *class*="login-form-field" *placeholder*="Username">

<input *type*="password" *name*="password" *id*="password-field" *class*="login-form-field" *placeholder*="Password">

<input *type*="submit" *value*="Login" *id*="login-form-submit">

</form>

</main>

</body>

<style>

html {

height: 100%;

}

body {

height: 100%;

margin: 0;

font-family: Arial, Helvetica, sans-serif;

display: grid;

justify-items: center;

align-items: center;

background-color: #3a3a3a;

}

*#logout* {

opacity: 0;

}

*#main-holder* {

width: 50%;

height: 70%;

display: grid;

justify-items: center;

align-items: center;

background-color: white;

border-radius: 7px;

box-shadow: 0px 0px 5px 2px black;

}

*#login-error-msg-holder* {

width: 100%;

height: 100%;

display: grid;

justify-items: center;

align-items: center;

}

*#login-error-msg* {

width: 23%;

text-align: center;

margin: 0;

padding: 5px;

font-size: 12px;

font-weight: bold;

color: #8a0000;

border: 1px solid #8a0000;

background-color: #e58f8f;

opacity: 0;

}

*#error-msg-second-line* {

display: block;

}

*#login-form* {

align-self: flex-start;

display: grid;

justify-items: center;

align-items: center;

}

*.login-form-field::placeholder* {

color: #3a3a3a;

}

*.login-form-field* {

border: none;

border-bottom: 1px solid #3a3a3a;

margin-bottom: 10px;

border-radius: 3px;

outline: none;

padding: 0px 0px 5px 5px;

}

*#login-form-submit* {

width: 100%;

padding: 7px;

border: none;

border-radius: 5px;

color: white;

font-weight: bold;

background-color: #3a3a3a;

cursor: pointer;

outline: none;

}

</style>

<script>

const session = sessionStorage.getItem('session');

let token;

*try* {

token = JSON.parse(session).access\_token;

} *catch*(e) {}

*if* (token) {

axios.get('/', {

headers: {

Authorization: token

}

}).then((response) => {

const { username } = response.data;

*if* (username) {

const mainHolder = document.getElementById("main-holder");

const loginHeader = document.getElementById("login-header");

loginForm.remove();

loginErrorMsg.remove();

loginHeader.remove();

mainHolder.append(`Hello ${username}`);

logoutLink.style.opacity = 1;

}

});

}

const loginForm = document.getElementById("login-form");

const loginButton = document.getElementById("login-form-submit");

const loginErrorMsg = document.getElementById("login-error-msg");

const logoutLink = document.getElementById("logout");

logoutLink.addEventListener("click", (e) => {

e.preventDefault();

sessionStorage.removeItem('session');

location.reload();

});

loginButton.addEventListener("click", (e) => {

e.preventDefault();

const login = loginForm.login.value;

const password = loginForm.password.value;

axios({

method: 'post',

url: '/api/login',

data: {

login,

password

}

}).then((response) => {

const { username } = response.data;

sessionStorage.setItem('session', JSON.stringify(response.data));

location.reload();

}).catch((response) => {

loginErrorMsg.style.opacity = 1;

});

})

</script>

</html>

Index.js

const uuid = require('uuid');

const express = require('express');

const onFinished = require('on-finished');

const bodyParser = require('body-parser');

const path = require('path');

const port = 3000;

const fs = require('fs');

const axios = require('axios');

const jwt = require('jsonwebtoken');

const jwksClient = require('jwks-rsa');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

const SESSION\_KEY = 'Authorization';

const AUTH0\_DOMAIN = 'dev-6sww0yh4s3mew71l.us.auth0.com';

const CLIENT\_ID = 'bpgWdV1Dlbin2T2VYq3J0nmsRe7zrZ5G';

const CLIENT\_SECRET = '1eJqaL3x\_EKq-\_682T6vY5KEKWKLWMK77l5R1WAc02CYvbAI7oUia49C02gRWiFf';

const AUDIENCE = 'https://dev-6sww0yh4s3mew71l.us.auth0.com/api/v2/';

const TOKEN\_URL = `https://${AUTH0\_DOMAIN}/oauth/token`;

const client = jwksClient({

jwksUri: `https://${AUTH0\_DOMAIN}/.well-known/jwks.json`

});

const getKey = (header, callback) => {

client.getSigningKey(header.kid, (err, key) => {

*if* (err) *return* callback(err);

const signingKey = key.publicKey || key.rsaPublicKey;

callback(null, signingKey);

});

};

class Session {

#sessions = {}

constructor() {

*try* {

*this*.#sessions = fs.readFileSync('./sessions.json', 'utf8');

*this*.#sessions = JSON.parse(*this*.#sessions.trim());

} *catch* (e) {

*this*.#sessions = {};

}

}

#storeSessions() {

fs.writeFileSync('./sessions.json', JSON.stringify(*this*.#sessions), 'utf-8');

}

set(key, value) {

*if* (!value) {

value = {};

}

*this*.#sessions[key] = value;

*this*.#storeSessions();

}

get(key) {

*return* *this*.#sessions[key];

}

init(res) {

const sessionId = uuid.v4();

*this*.set(sessionId);

*return* sessionId;

}

destroy(req, res) {

let sessionId = *this*.findSessionByAccessToken(req.session.access\_token);

*while*(sessionId){

console.log("hi", sessionId);

delete *this*.#sessions[sessionId];

sessionId = *this*.findSessionByAccessToken(req.session.access\_token);

}

*this*.#storeSessions();

}

findSessionByAccessToken(accessToken) {

*for* (const sessionId in *this*.#sessions) {

*if* (*this*.#sessions[sessionId].access\_token === accessToken) {

*return* *this*.#sessions[sessionId];

}

}

*return* null;

}

getSessionFromAccessTokenOrCreate(accessToken, res) {

let currentSession = *this*.findSessionByAccessToken(accessToken);

let sessionId;

*if* (currentSession) {

sessionId = currentSession.sessionId;

}

*return* { currentSession, sessionId };

}

}

const sessions = new Session();

app.use((req, res, next) => {

let currentSession = {};

const accessToken = req.get(SESSION\_KEY);

let sessionId = req.sessionId;

*if* (accessToken) {

console.log("Trying to find session by access\_token");

jwt.verify(accessToken, getKey, { algorithms: ['RS256'] }, (err, decoded) => {

*if* (err) {

*return* res.status(401).json({ error: 'Unauthorized' });

}

const { currentSession, sessionId } = sessions.getSessionFromAccessTokenOrCreate(accessToken, res);

req.session = currentSession;

req.sessionId = sessionId;

next();

});

} *else* {

*if* (sessionId) {

currentSession = sessions.get(sessionId);

} *else* {

sessionId = sessions.init(res);

}

req.session = currentSession;

req.sessionId = sessionId;

next();

}

onFinished(req, () => {

const currentSession = req.session;

const sessionId = req.sessionId;

sessions.set(sessionId, currentSession);

});

});

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

*if* (req.session.username) {

*return* res.json({

username: req.session.username,

logout: 'http://localhost:3000/logout'

});

}

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

});

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

sessions.destroy(req, res);

res.redirect('/');

});

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

const { login, password } = req.body;

*try* {

const response = *await* axios.post(TOKEN\_URL, {

grant\_type: 'password',

username: login,

password: password,

audience: AUDIENCE,

client\_id: CLIENT\_ID,

client\_secret: CLIENT\_SECRET,

});

const { access\_token} = response.data;

req.session.username = login;

req.session.access\_token = access\_token;

*return* res.json({

message: 'Login successful',

access\_token: access\_token,

});

} *catch* (error) {

console.error('Error during Auth0 login:', error.response?.data || error.message);

res.status(401).json({

error: 'Invalid login credentials or unauthorized.'

});

}

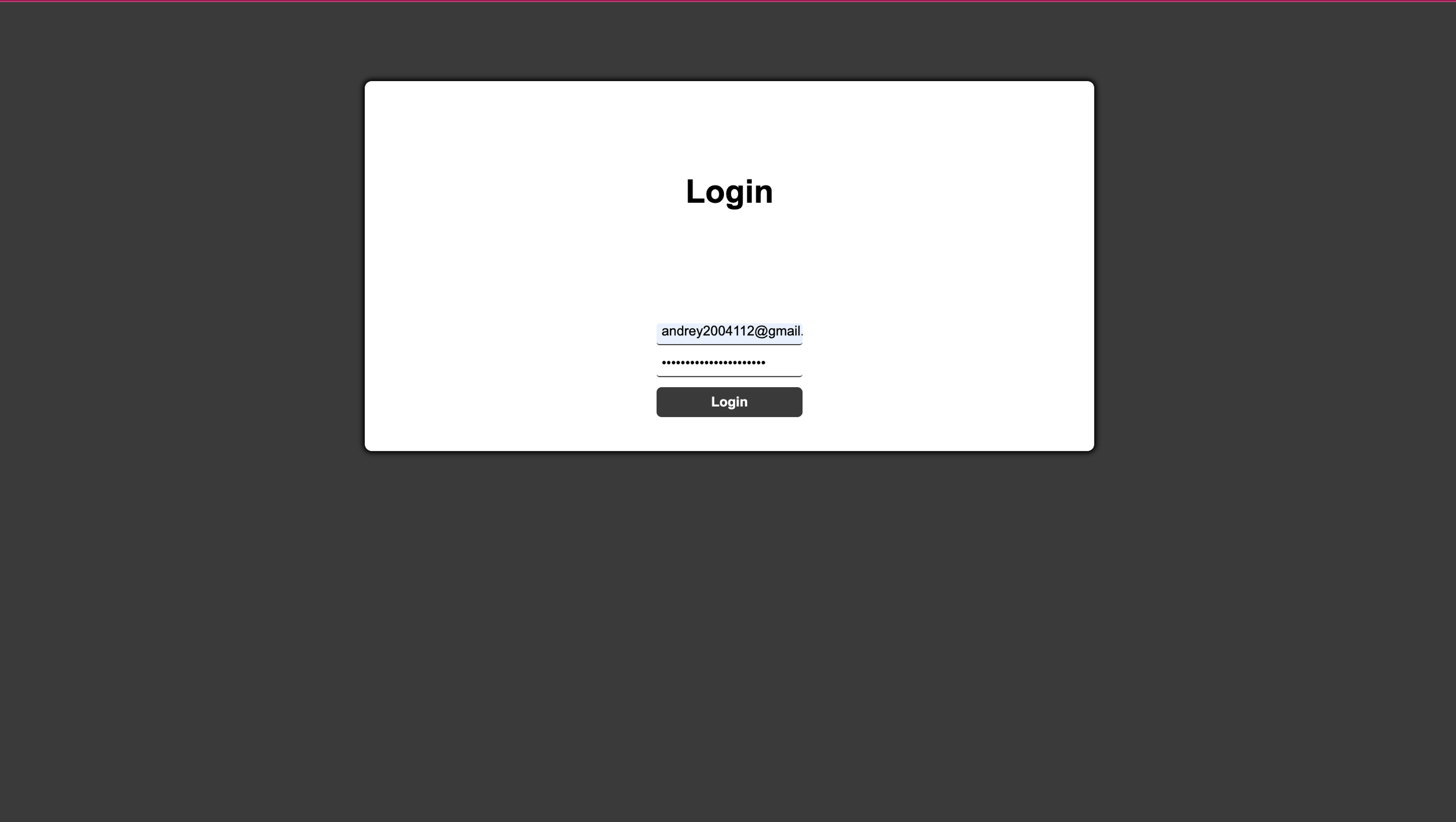
});

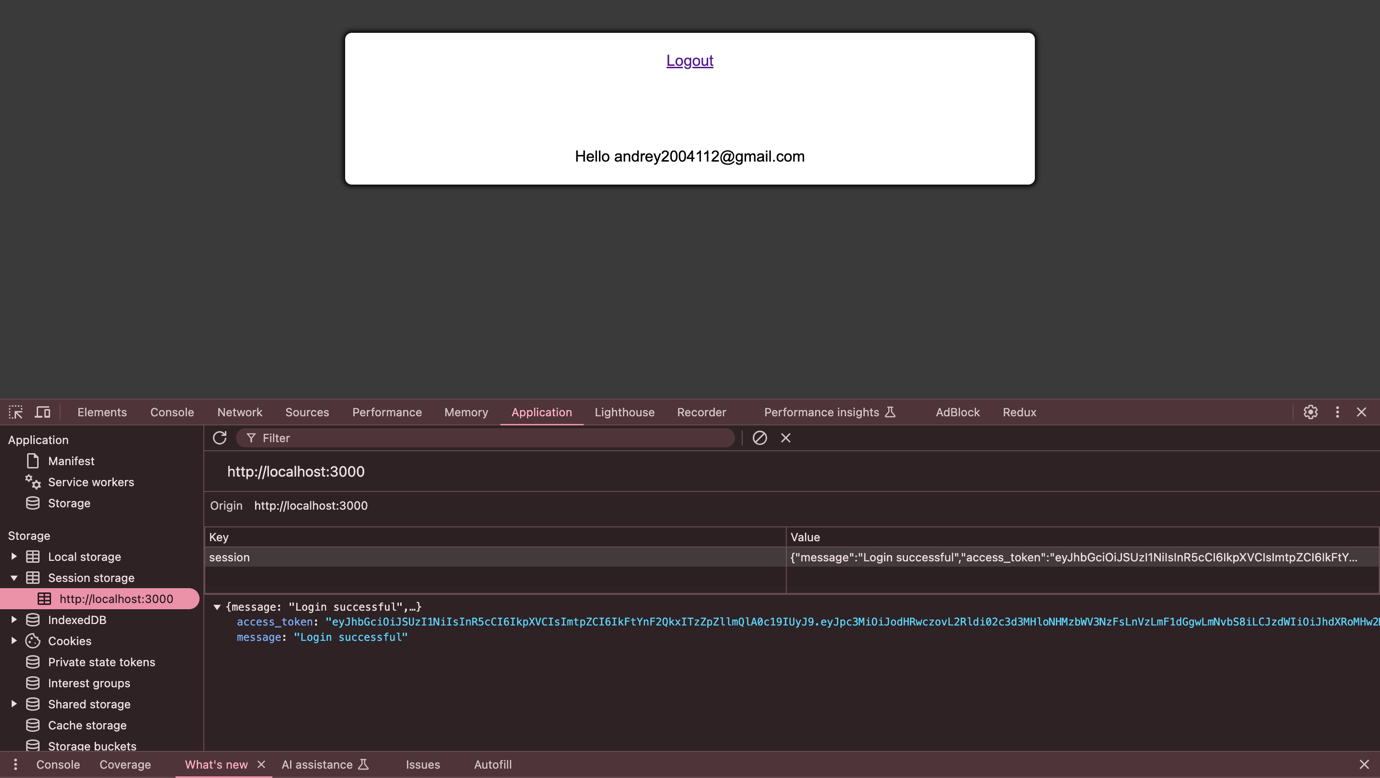
app.listen(port, () => {

console.log(`Example app listening on port ${port}`);

});

Результат:





## ВИСНОВКИ

В результаті виконання лабораторної роботи було досягнуто наступних результатів:

**Додавання валідації JWT токенів**:

Було успішно реалізовано верифікацію **JWT** (JSON Web Token) токенів на серверній стороні. Для цього використано бібліотеку jsonwebtoken, а також налаштування **JWKS (JSON Web Key Set)** для отримання публічного ключа, необхідного для верифікації токена при асиметричному шифруванні.

У коді було використано функцію jwt.verify() для перевірки підпису отриманого токена та забезпечення його дійсності.

**Імплементація збереження сесій**:

Була реалізована система сесій, яка зберігає інформацію про користувача та його токен доступу. За допомогою сесійного ідентифікатора в серверному коді організовано зберігання даних сесій у JSON-файлі.

За допомогою цього підходу забезпечено збереження авторизаційних даних на сервері між запитами користувачів.

**Аутентифікація через Auth0**:

Для аутентифікації користувачів було використано сервіс **Auth0**, який надає можливість отримання токенів доступу через стандартний процес авторизації.

Після отримання токена через **Auth0**, цей токен був використаний для авторизації користувача та його валідації на сервері.

**Робота з фронтендом**:

У HTML-формі було реалізовано введення даних користувача (логін і пароль), після чого дані передаються на сервер для аутентифікації.

В результаті успішної аутентифікації на сервері, у браузері зберігається сесія користувача у sessionStorage, що дозволяє підтримувати авторизацію між сторінками без необхідності повторно вводити логін і пароль.

**Використання асиметричного шифрування**:

Для верифікації JWT токена в разі використання асиметричного шифрування використовувався публічний ключ, отриманий через посилання на API для Auth0.

Завдяки використанню публічного ключа, було забезпечено можливість перевіряти правильність підпису JWT токенів, що дозволяє захистити систему від підроблених токенів.