



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

Лабораторна робота №1

Безпека ПЗ

Тема: Огляд основних методів авторизації

Виконав

студент групи ІП-11:

Панченко С. В.

Перевірив:

Курченко О. А.

Київ 2024

ЗМІСТ

1 Мета комп'ютерного практикуму.....	6
2 Завдання.....	7
3 Виконання.....	8
3.1 Огляд basic_auth.....	8
3.2 Огляд forms_auth.....	8
3.3 Огляд token_auth.....	12
Висновок.....	13
ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ.....	14

1 МЕТА КОМП'ЮТЕРНОГО ПРАКТИКУМУ

Роздивитися основні методи авторизації.

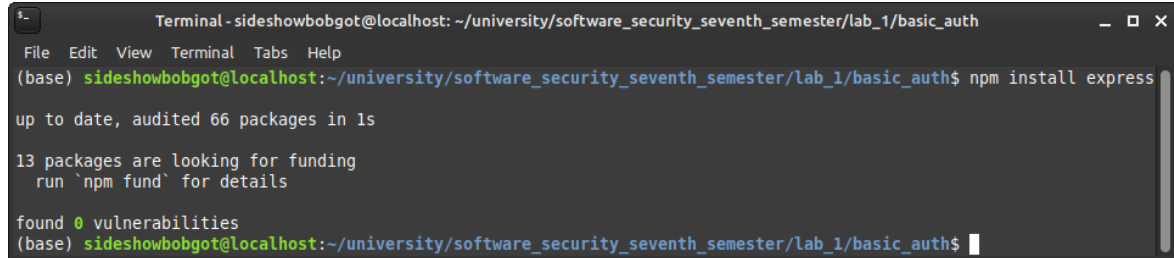
2 ЗАВДАННЯ

1. Викачати репозиторій з лекціями https://github.com/Kreolwolf1/auth_examples. Запустити кожен з 3 аплікейшенів та зробити скріншоти запитів до серверу.
2. Модифікувати token_auth аплікейшен змінивши токен на JWT.

3 ВИКОНАННЯ

3.1 Огляд basic_auth

Встановимо фреймворк express на рисунку 3.1.



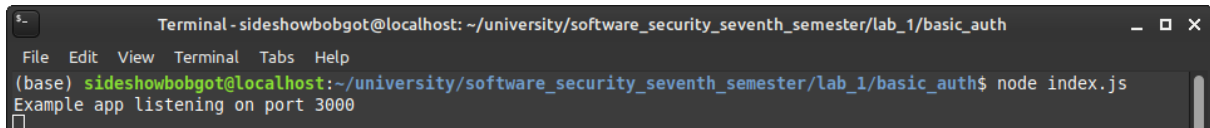
```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semester/lab_1/basic_auth
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/lab_1/basic_auth$ npm install express
up to date, audited 66 packages in 1s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/lab_1/basic_auth$
```

Рисунок 3.1 — Встановлення express

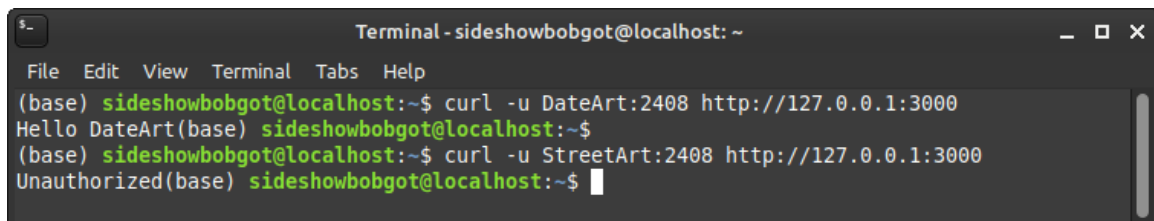
Запустимо сервер. Він слухатиме нас на порті 3000. Розглянемо ці кроки на рисунку 3.2.



```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semester/lab_1/basic_auth
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/lab_1/basic_auth$ node index.js
Example app listening on port 3000
```

Рисунок 3.2 — Запуск сервера

Спробуємо звернутися до сервера з логіном DateArt та паролем 2048. Побачимо, що авторизація успішна. Після цього звернемося до сервера з іншим логіном. Побачимо, що авторизація не є успішною. Розглянемо ці кроки на рисунку 3.3.

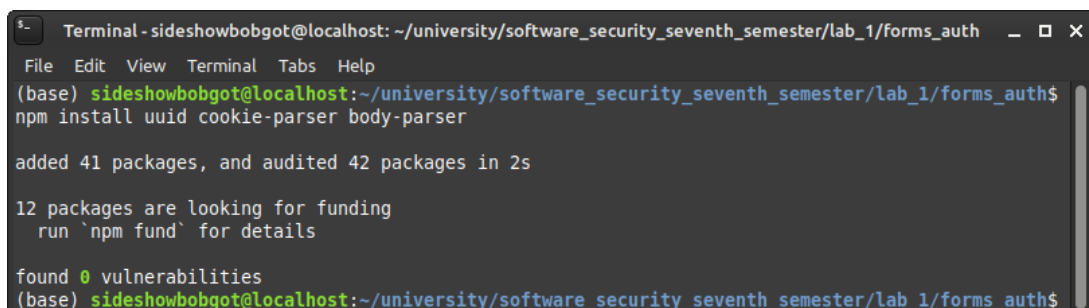


```
Terminal - sideshowbobgot@localhost: ~
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~$ curl -u DateArt:2408 http://127.0.0.1:3000
Hello DateArt(base) sideshowbobgot@localhost:~$
(base) sideshowbobgot@localhost:~$ curl -u StreetArt:2408 http://127.0.0.1:3000
Unauthorized(base) sideshowbobgot@localhost:~$
```

Рисунок 3.3 — Приклад успішної та неуспішної авторизації

3.2 Огляд forms_auth

Встановимо npm-пакети, запустимо сервер на рисунку 3.4.



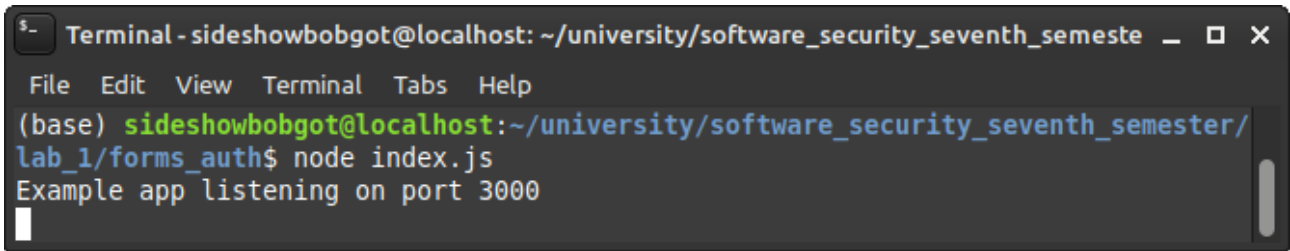
```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semester/lab_1/forms_auth
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/lab_1/forms_auth$ npm install uuid cookie-parser body-parser
added 41 packages, and audited 42 packages in 2s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/lab_1/forms_auth$
```

Рисунок 3.4 — Встановлення npm-пакетів

Запустимо сервер на рисунку 3.5.



```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semeste _ □ ×
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/
lab_1/forms_auth$ node index.js
Example app listening on port 3000
```

Рисунок 3.5 — Запуск сервера

Зайдемо до браузера за адресою <http://127.0.0.1:3000> на рисунку 3.6.

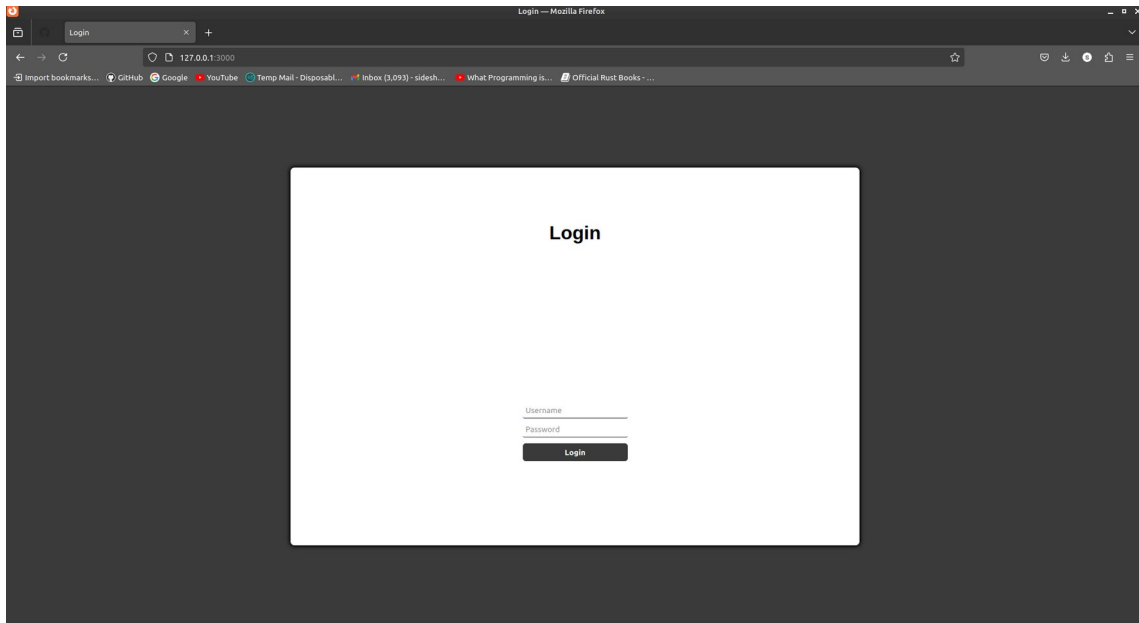


Рисунок 3.6 — Сторінка за замовчуванням

Уводимо логін(Login) та пароль>Password) на рисунку 3.7. Бачимо, що авторизація успішна.

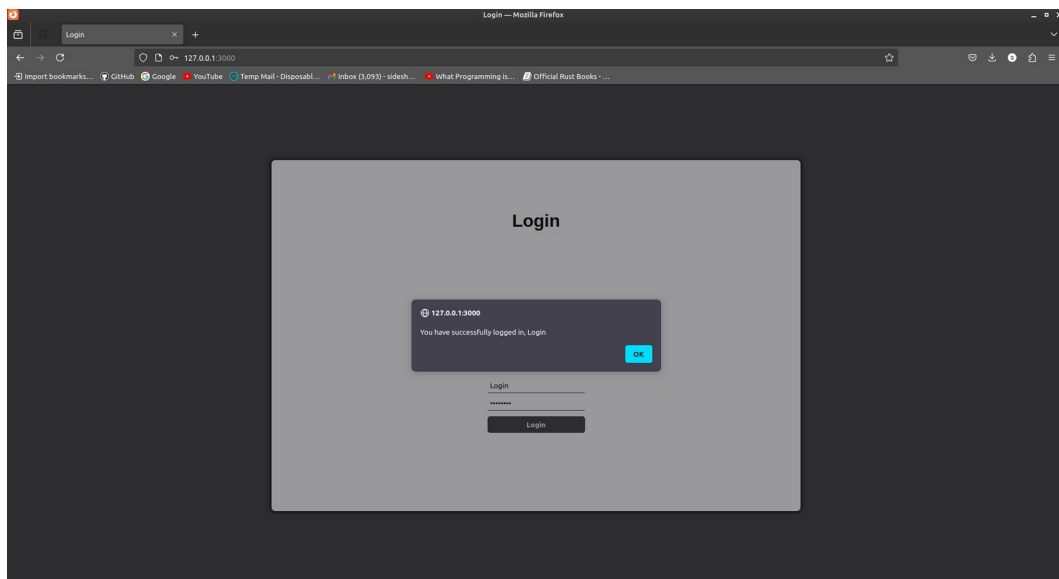


Рисунок 3.7 — Успішна авторизація

Розглянемо запити через DevTools у Mozilla Firefox на рисунку 3.8.

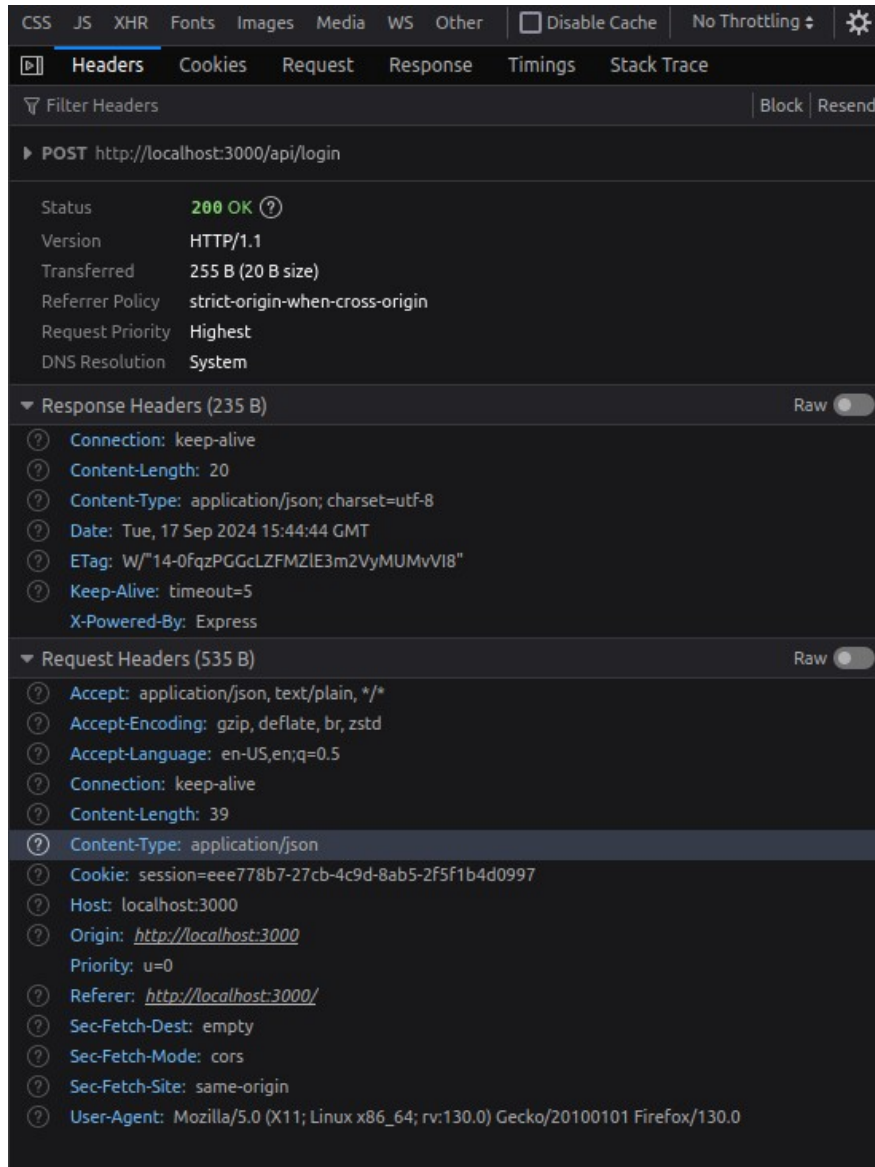


Рисунок 3.8 — POST-запит авторизації

Бачимо, що це є Post запит. На рисунку 3.9 розглянемо payload.

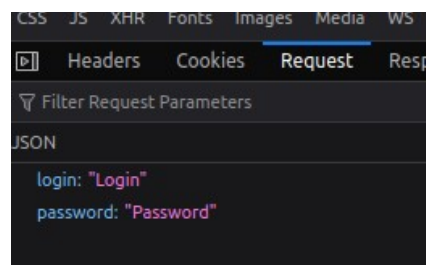


Рисунок 3.9 — Payload запиту

Розглянемо файл `sessions.json`. Бачимо, що дані передається сирим JSON. Бачимо, що сесія запиту та `session.json` збігаються на рисунку 3.10 та 3.11 відповідно.

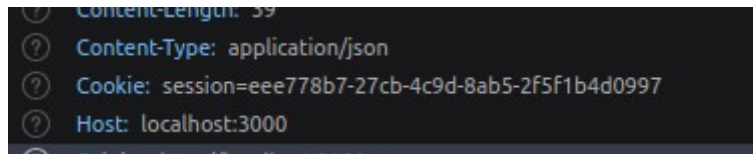


Рисунок 3.10 — Сесія запиту



Рисунок 3.11 — Сесія sessions.json

Вийдемо з сесії. Бачимо, що сервер видаляє сесію, повертаючи заголовок Set-Cookie на рисунку 3.12. Але видалення не відбувається з файлу sessions.json.

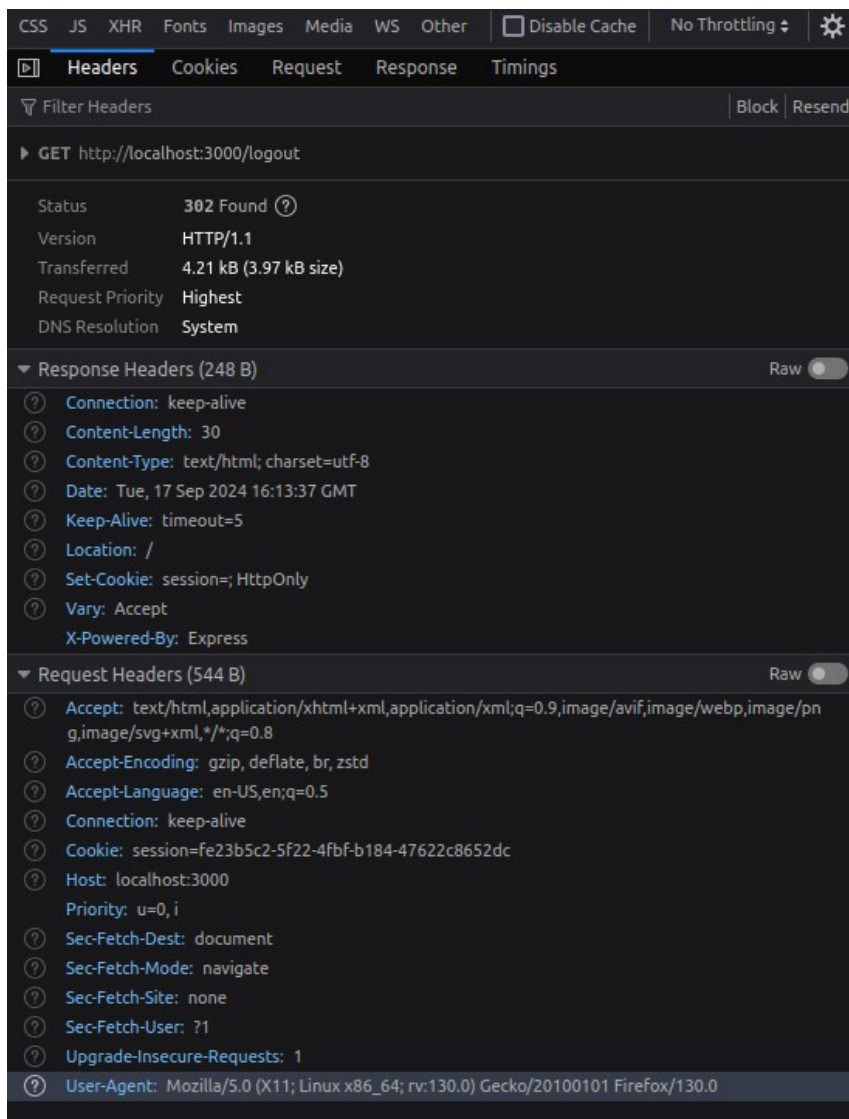
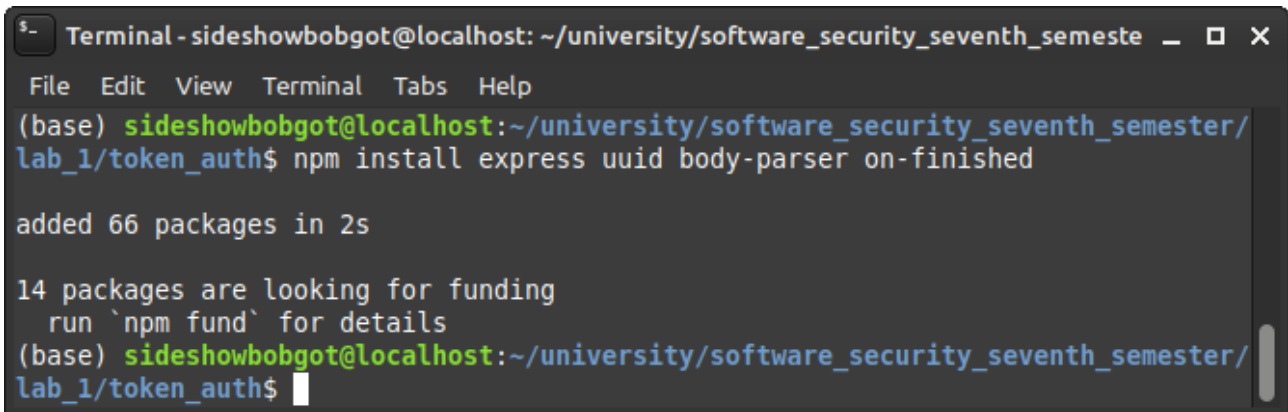


Рисунок 3.12 — Вихід з сесії

3.3 Огляд token_auth

Встановлюємо залежності та запускаємо сервер на рисунку 3.13.



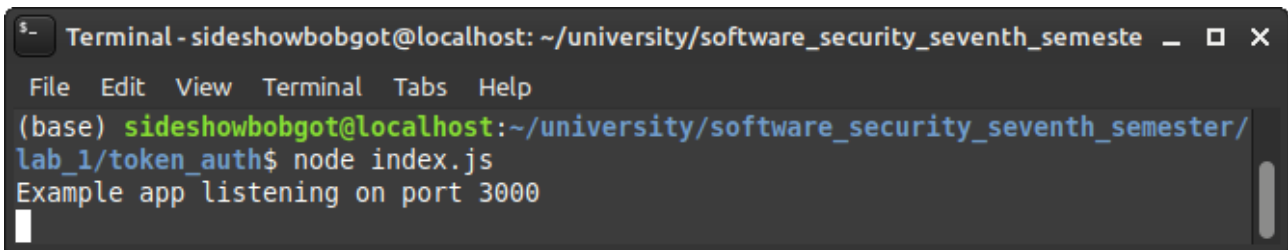
```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semeste
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/
lab_1/token_auth$ npm install express uuid body-parser on-finished

added 66 packages in 2s

14 packages are looking for funding
  run `npm fund` for details
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/
lab_1/token_auth$
```

Рисунок 3.13 — Встановлення пакетів

Запускаємо сервер на рисунку 3.14.



```
Terminal - sideshowbobgot@localhost: ~/university/software_security_seventh_semeste
File Edit View Terminal Tabs Help
(base) sideshowbobgot@localhost:~/university/software_security_seventh_semester/
lab_1/token_auth$ node index.js
Example app listening on port 3000
```

Рисунок 3.14 — Запуск сервера

Авторизуємося до сервера з Login та Password на рисунку 3.15. Бачимо, що нам відображається Username.

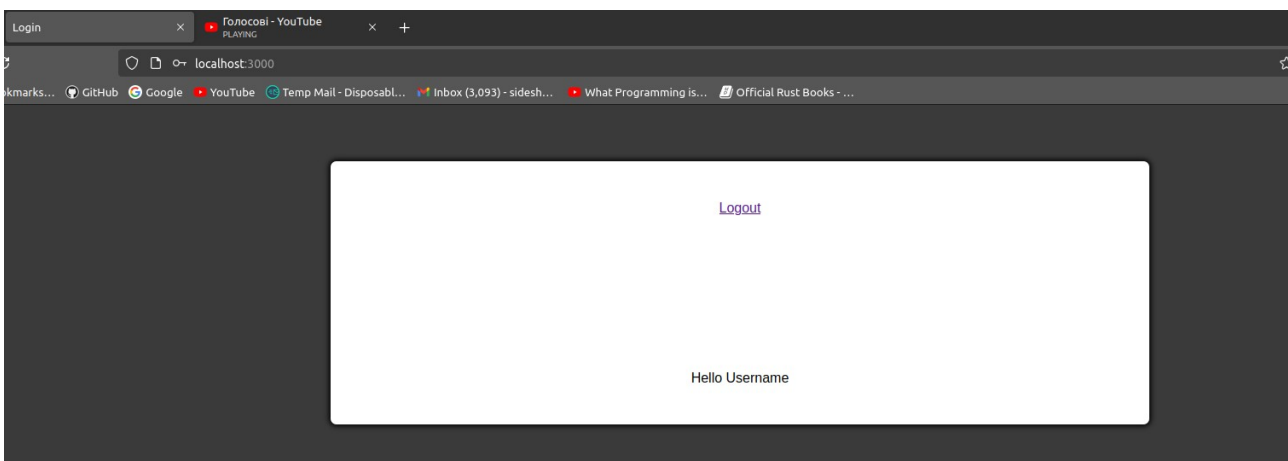


Рисунок 3.16 — Сторінка після авторизації з Login та Password.

Авторизуємося до сервера з Login1 та Password1 на рисунку 3.16. Бачимо, що нам відображається Username1.

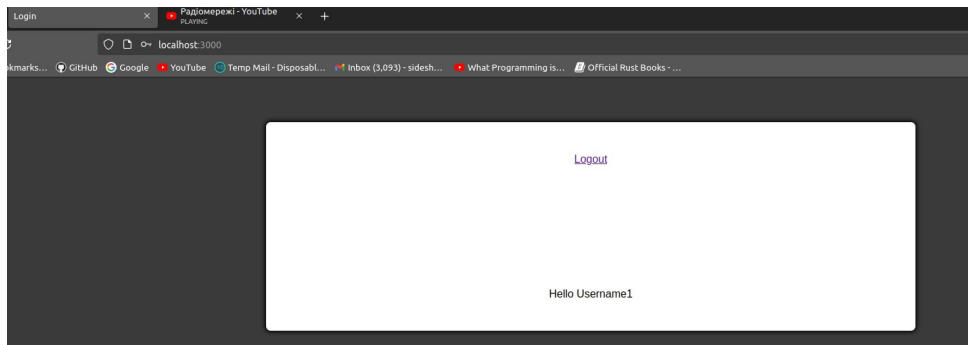


Рисунок 3.17 — Сторінка після авторизації з Login1 та Password1.

Спробуємо авторизуватися з неправильним логіном та паролем на рисунку 3.18.

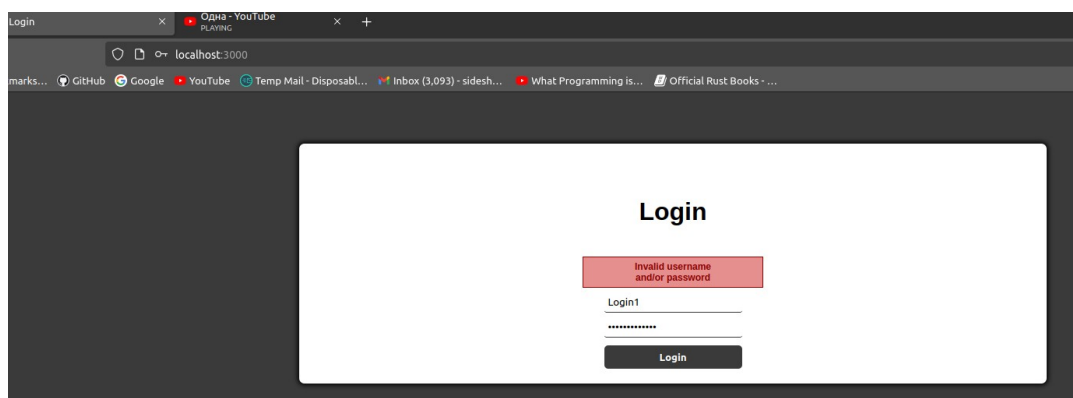


Рисунок 3.18 — Невдала спроба авторизації.

3.4 Додаткове завдання

Змінимо файли `index.html` та `index.js`. Код наведено у додатках.

На рисунку 3.19 бачимо сторінку за замовчуванням.

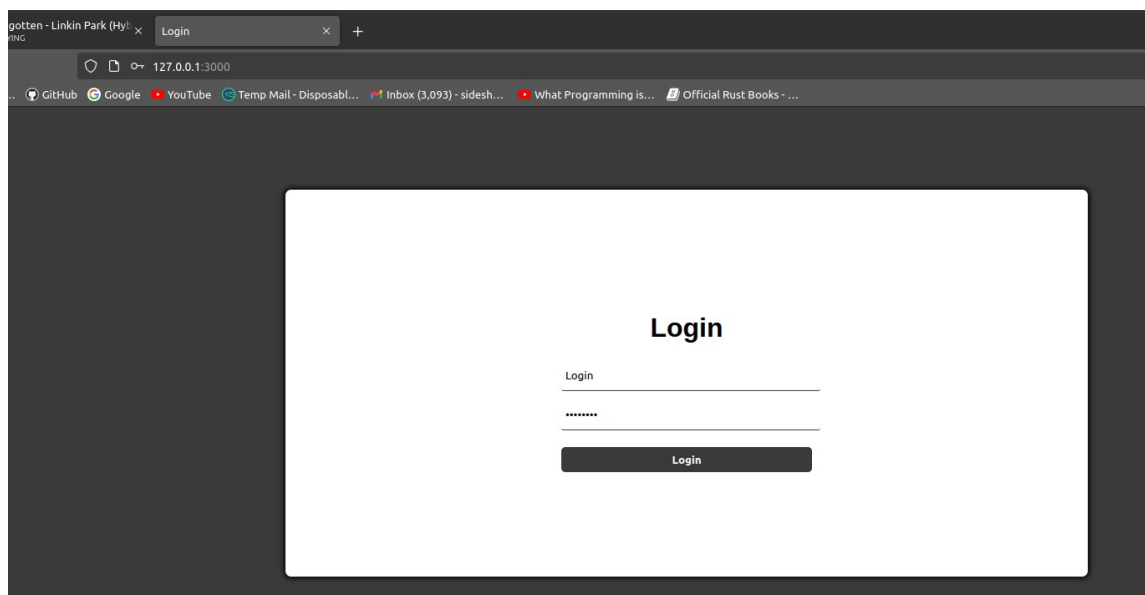


Рисунок 3.19 — Сторінка за замовчуванням

Вводимо Login та Password на рисунку 3.20 та бачимо, що у хедері Authorization маємо Bearer <token> на рисунку 3.21.

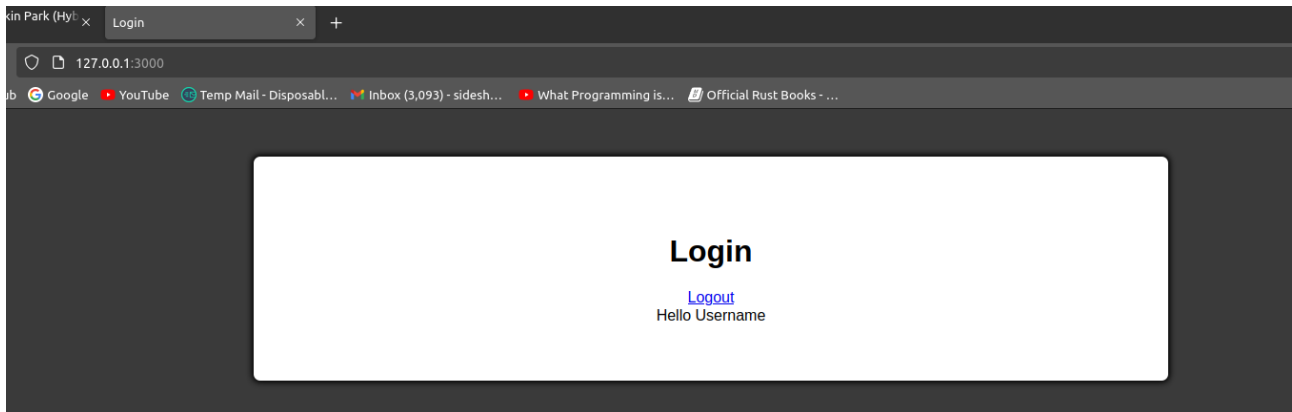


Рисунок 3.20 — Успішний вхід до сторінки

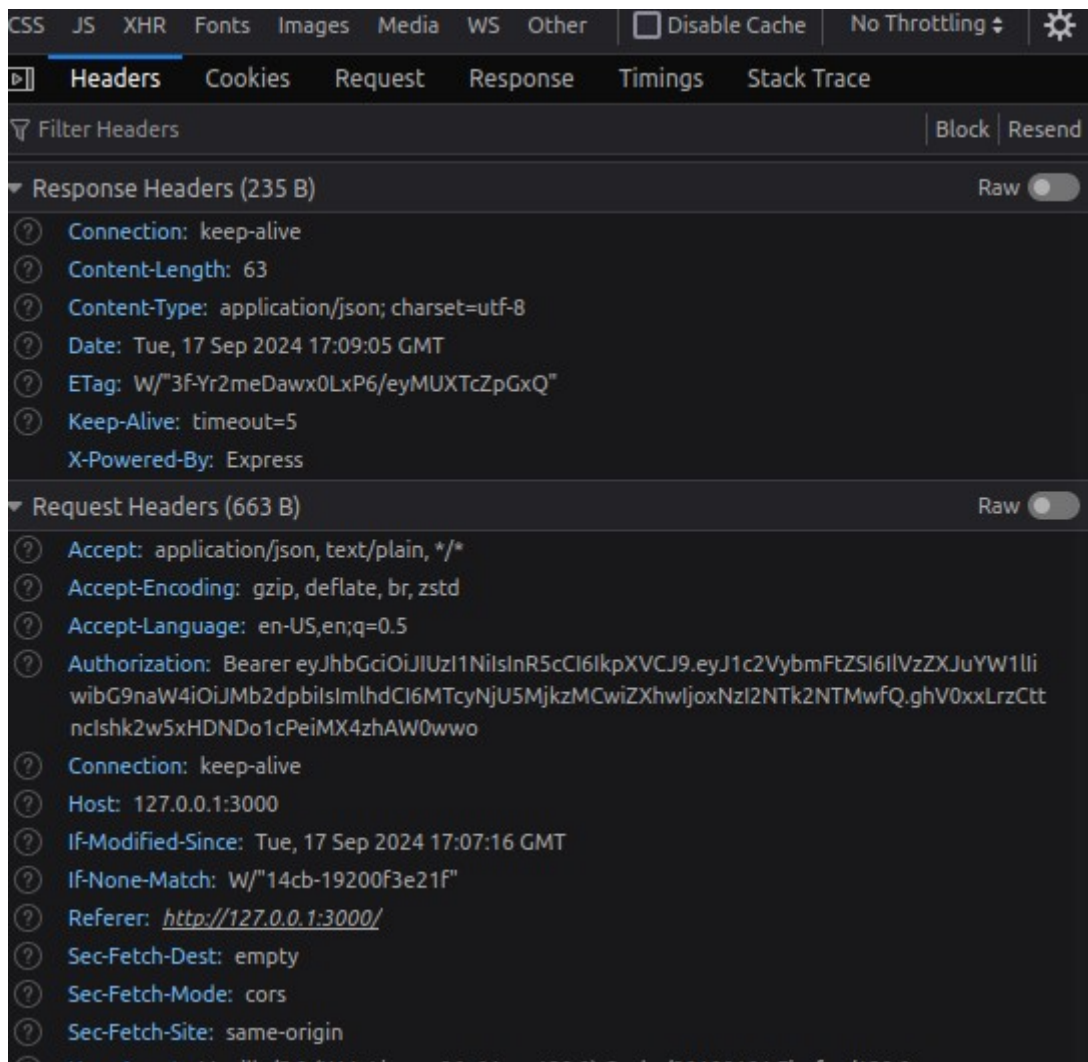


Рисунок 3.21 — Токен

ВИСНОВОК

В процесі виконання цієї лабораторної роботи ми дослідили кілька основних методів авторизації. Спочатку ми вивчили прямий метод, де авторизаційна інформація включалася безпосередньо в запити. Далі ми розглянули підхід з використанням ідентифікатора сесії, який ми отримували, надсилаючи дані у форматі JSON, і зберігали в куках браузера. Наступним кроком було вивчення методу, де токен авторизації, отриманий аналогічним способом, зберігався вже у localStorage. Насамкінець, ми модифікували останній метод, впровадивши технологію JWT (JSON Web Token) для авторизації.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду
(Найменування програми (документа))

Жорсткий диск
(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-11 4 курсу
Панченка С. В

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>
  <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;
    }
    #main-holder {
      width: 50%;
      height: 70%;
      display: flex;
      flex-direction: column;
      justify-content: 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 {
        display: flex;
        flex-direction: column;
        align-items: center;
        width: 100%;
        max-width: 300px;
    }
    .login-form-field::placeholder {
        color: #3a3a3a;
    }
    .login-form-field {
        width: 100%;
        border: none;
        border-bottom: 1px solid #3a3a3a;
        margin-bottom: 10px;
        border-radius: 3px;
        outline: none;
        padding: 10px 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;
        margin-top: 10px;
    }
}
</style>

```

```

</head>
<body>
  <main id="main-holder">
    <h1 id="login-header">Login</h1>
    <div id="login-error-msg-holder" style="display: none;">
      <p id="login-error-msg">Invalid username and/or password</p>
    </div>
    <form id="login-form">
      <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>
    <a href="#" id="logout" style="display: none;">Logout</a>
  </main>

  <script>
    document.getElementById("login-form").addEventListener("submit",
function (e) {
      e.preventDefault();
      const login = document.getElementById("username-field").value;
      const password = document.getElementById("password-
field").value;
      axios.post('/api/login', {login, password}).then(response => {
        sessionStorage.setItem('authToken', response.data.token);
        alert('You are now logged in!');
        document.getElementById("logout").style.display = "block";
        document.getElementById("login-form").style.display =
"none";
      }).catch(error => {
        document.getElementById("login-error-msg-
holder").style.display = "block";
      });
    });

    document.getElementById("logout").addEventListener("click",
function (e) {
      e.preventDefault();
      sessionStorage.removeItem('authToken');
      alert('You are now logged out!');
      document.getElementById("login-form").style.display = "block";
      this.style.display = "none";
    });
  </script>

```



```

    if (sessionStorage.getItem('authToken')) {
        document.getElementById("login-form").style.display = "none";
        document.getElementById("logout").style.display = "block";
    }

    document.addEventListener("DOMContentLoaded", function () {
        const token = sessionStorage.getItem('authToken');
        if (token) {
            axios.get('/', {
                headers: {
                    'Authorization': `Bearer ${token}`
                }
            }).then(response => {
                const {username} = response.data;
                if (username) {
                    const mainHolder = document.getElementById("main-
holder");
                    const loginHeader =
document.getElementById("login-header");
                    mainHolder.append(`Hello ${username}`);
                    loginForm.remove();
                    loginErrorMsg.remove();
                    loginHeader.remove();
                    logoutLink.style.opacity = 1;
                }
            }).catch(error => {});
        }
    });
</script>
</body>
</html>

```

index.js

```

const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const path = require('path');
const port = 3000;
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

```

```

const JWT_SECRET_KEY = 'IP11_Panchenko';
const users = [
  {
    login: 'Login',
    password: 'Password',
    username: 'Username',
  },
  {
    login: 'Login1',
    password: 'Password1',
    username: 'Username1',
  }
];
app.post('/api/login', (req, res) => {
  const { login, password } = req.body;
  const user = users.find(user => user.login === login &&
    user.password === password);
  if (user) {
    const token = jwt.sign({ username: user.username,
      login: user.login }, JWT_SECRET_KEY, { expiresIn: '1h' });
    res.json({ token });
  } else {
    res.status(401).send('Unauthorized');
  }
});
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (token == null) {
    if (req.path === '/' || req.path === '/api/login')
      return next();
    else return res.sendStatus(401);
  }
  jwt.verify(token, JWT_SECRET_KEY, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
};
app.get('/', authenticateToken, (req, res) => {
  console.log(req.user);
  if (req.user) {
    return res.json({
      username: req.user.username,

```

```
        logout: 'http://localhost:3000/logout'
      })
    }
    res.sendFile(path.join(__dirname+'/index.html'));
  });
app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```