Міністерство освіти та науки України
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра моделювання та програмного забезпечення

Практична робота № 5

з дисципліни «Сучасні технології Internet-програмування»

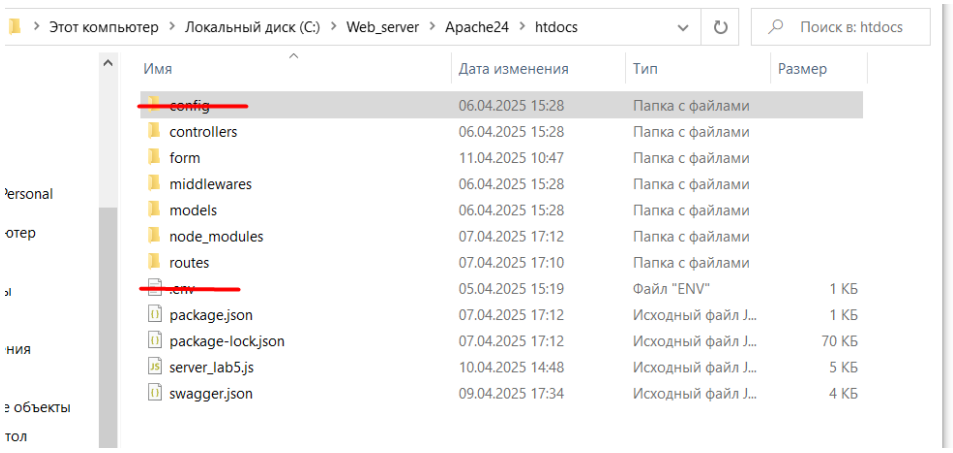
студент групи ІПЗ-22-1
Ющенко М.О.

Перевірили викладачі:
Трачук А.А.

Кривий Ріг
2025 р.

Завдання: 20. Створити API для дошки оголошень: CRUD для оголошень, авторизація JWT, WebSockets для миттєвого оновлення нових пропозицій.

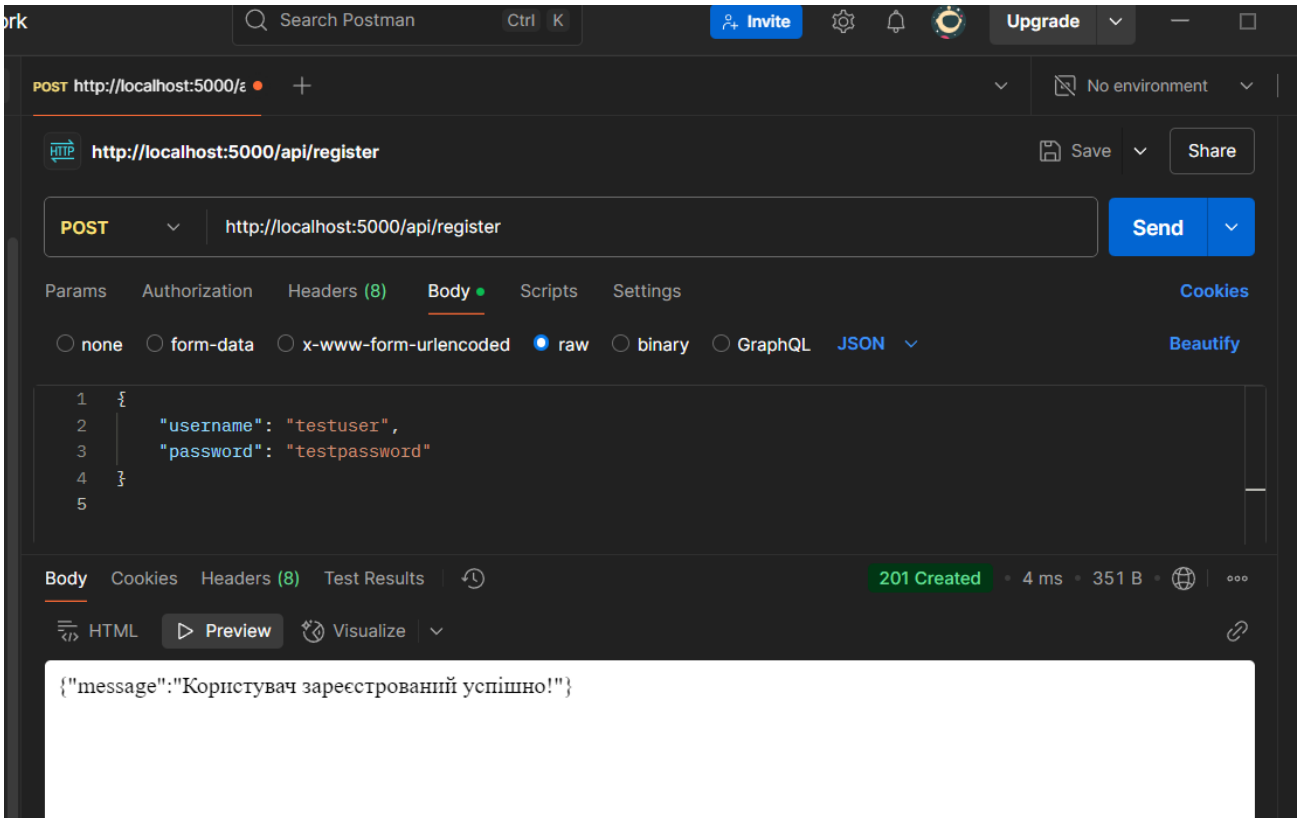
Папка з усіма файлами роботи. Передбачалося використання БД Mongo, але в процесі було прийняте рішення про зберігання даних у пам’яті браузера.



Встановимо потрібні бібліотеки.

Тестування серверу за допомогою POSTMAN:

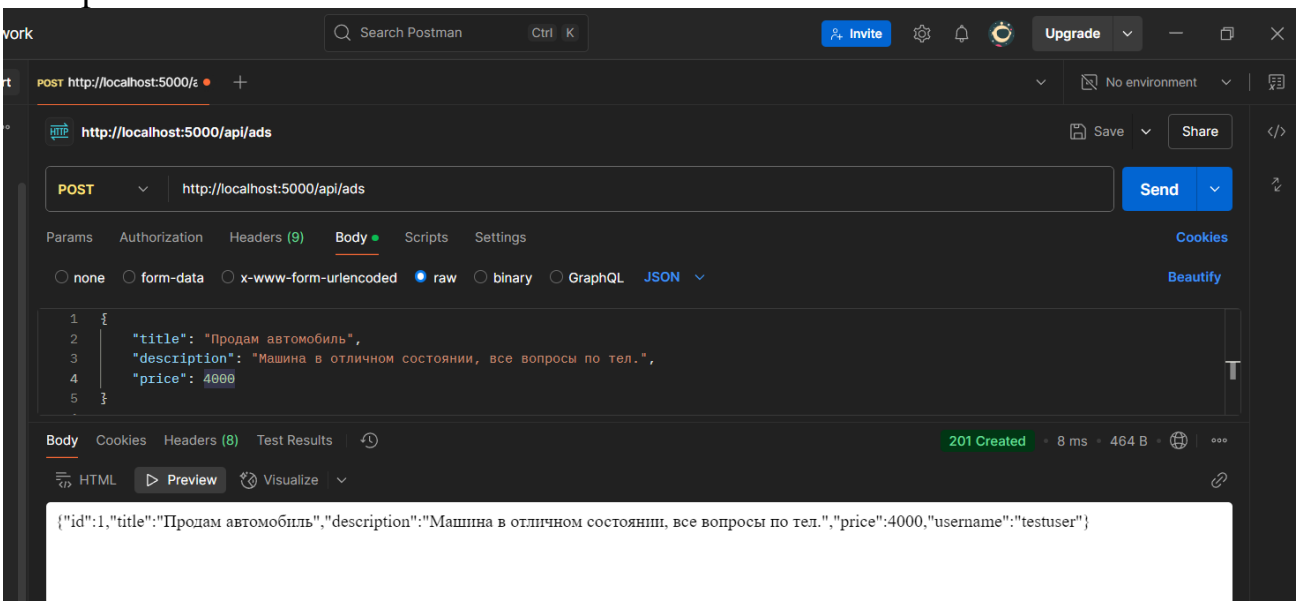
Протестуємо реєстрацію методом POST

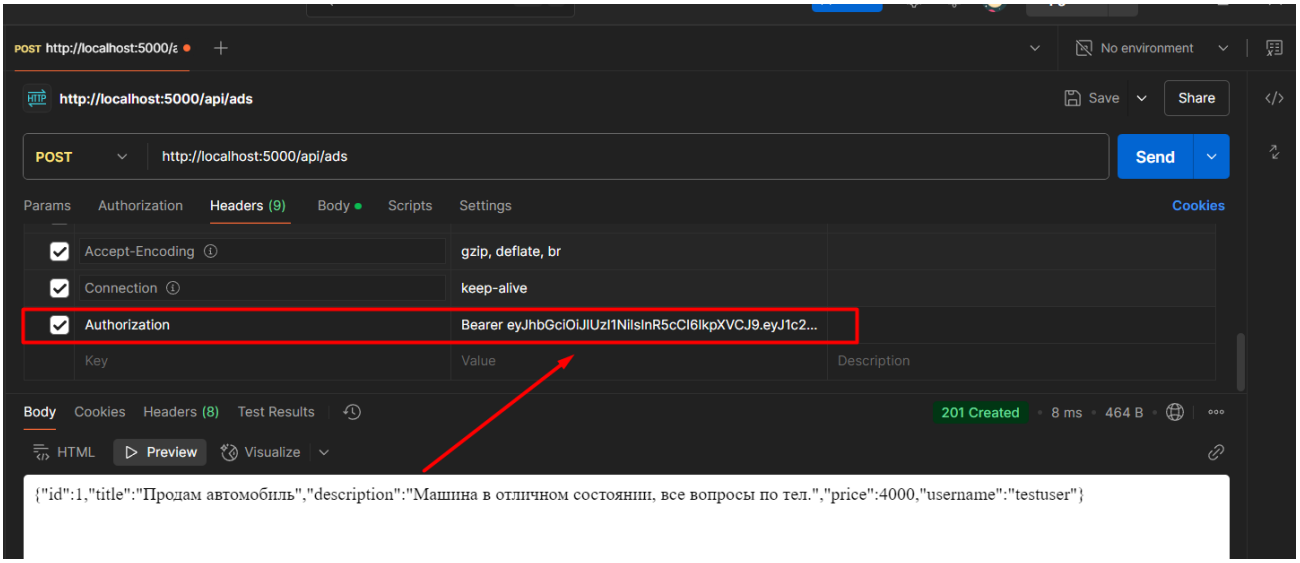


Отримали такий токен.

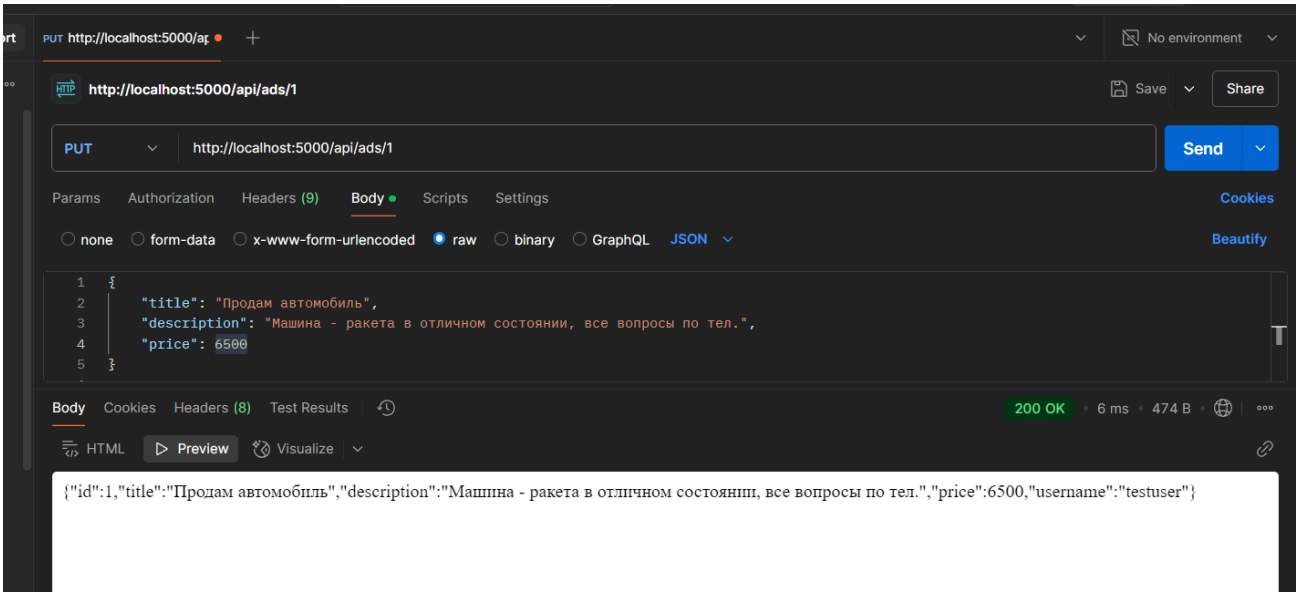
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3R1c2VyIiwiaWF0IjoxNzQ0MjEyNjAyfQ.ql5btK_WV4vca41EYVK04OEuwoAX3besI2TFUeZxAks"}

Створення оголошення:

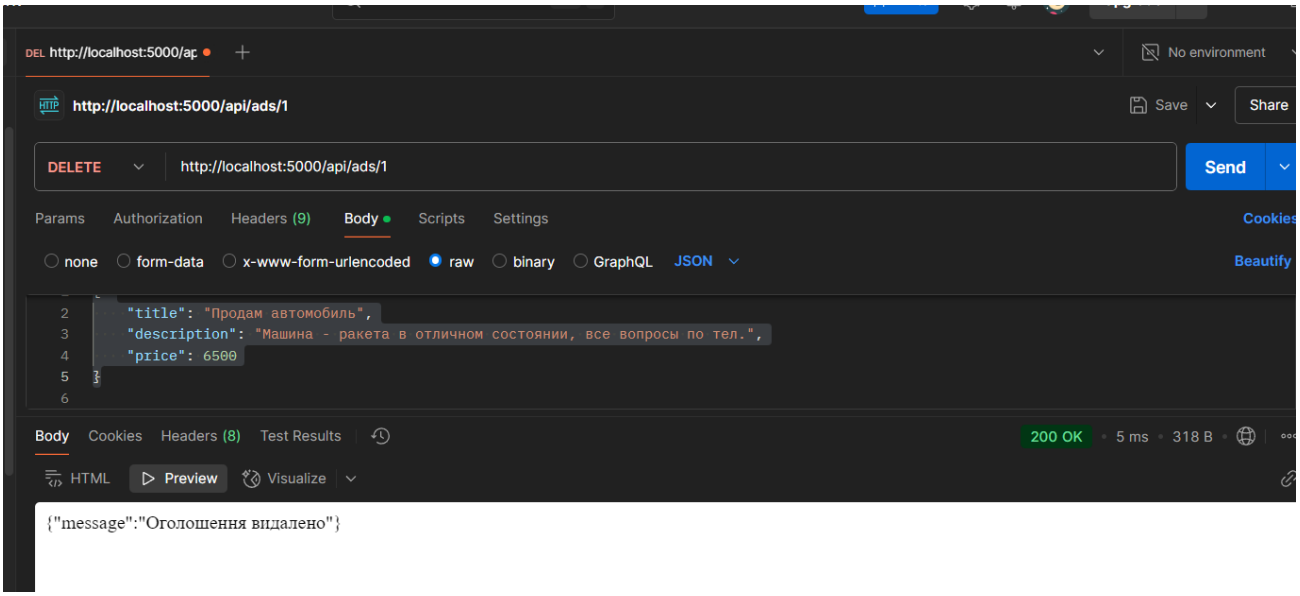




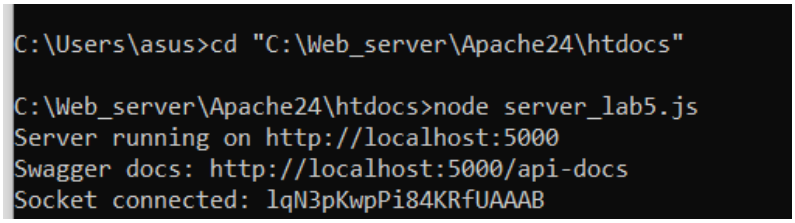
Зміна:



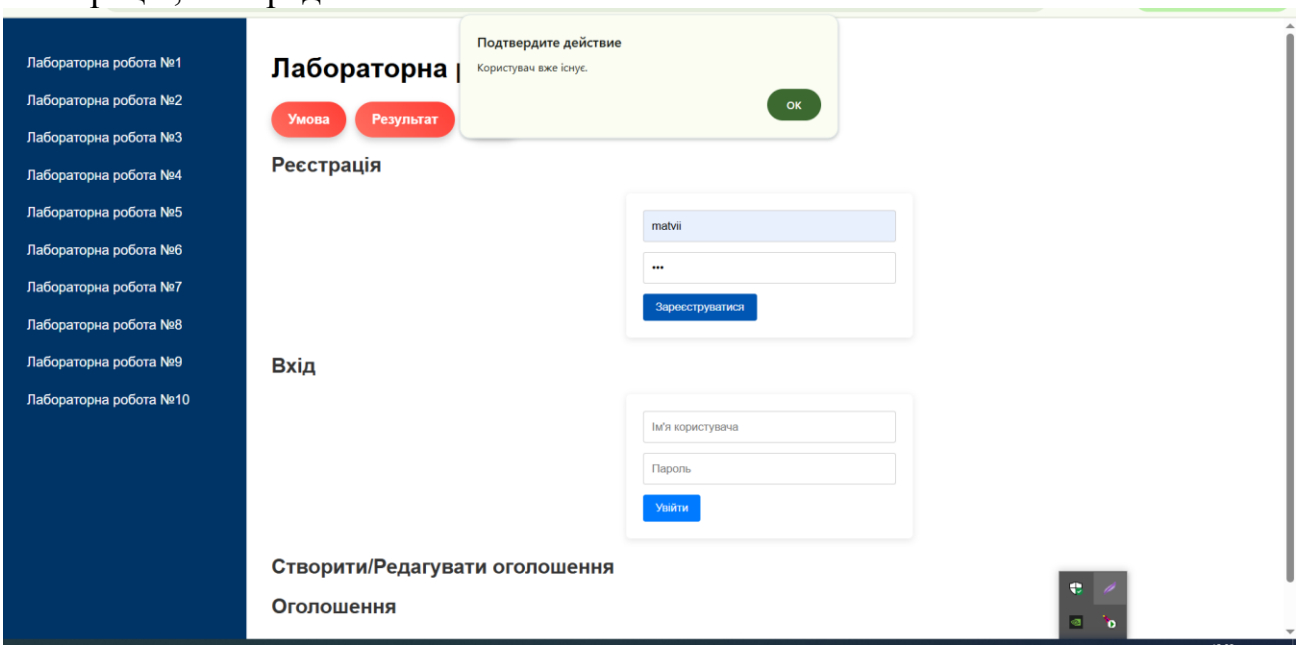
Видалення:



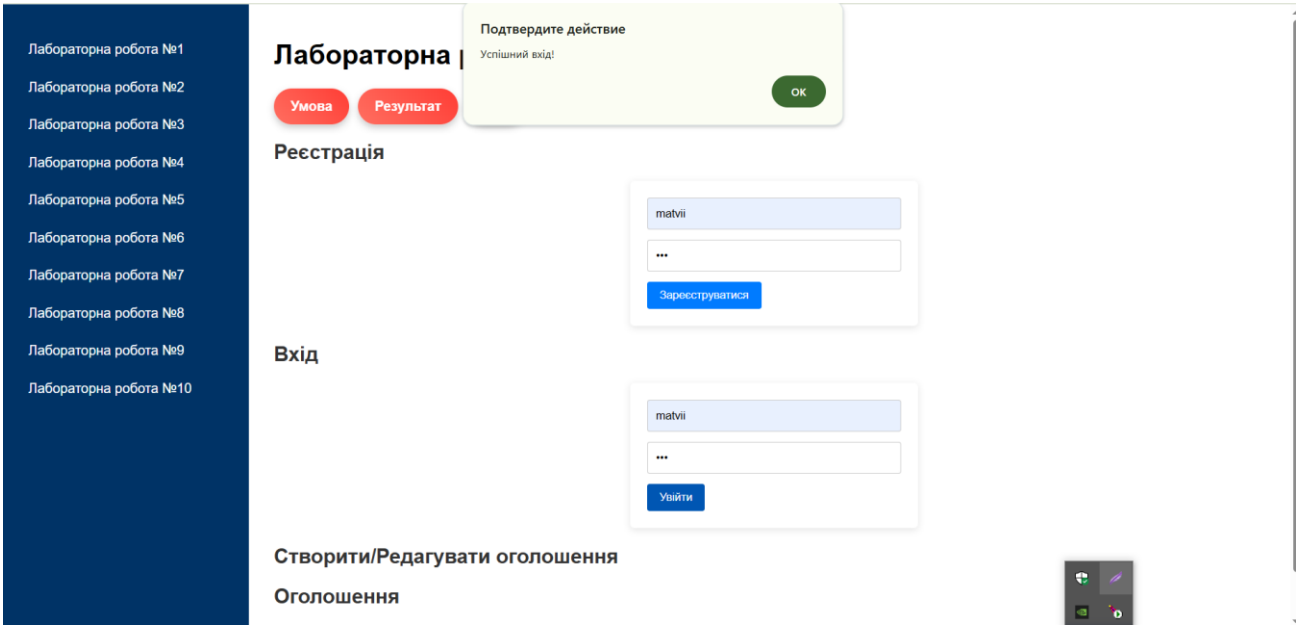
Скріншоти роботи сайту:
Запуск сервера.



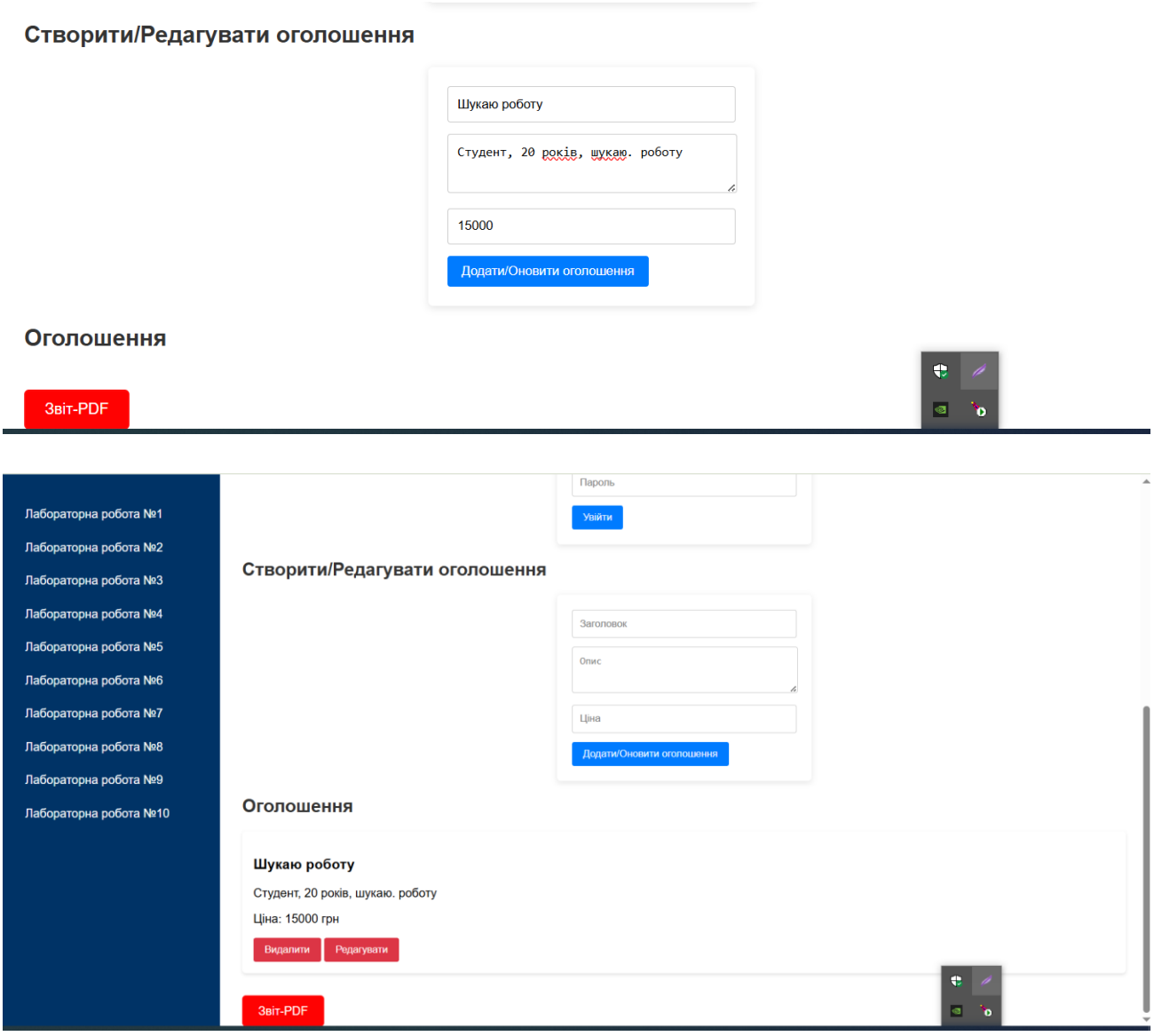
Реєстрація, попередження



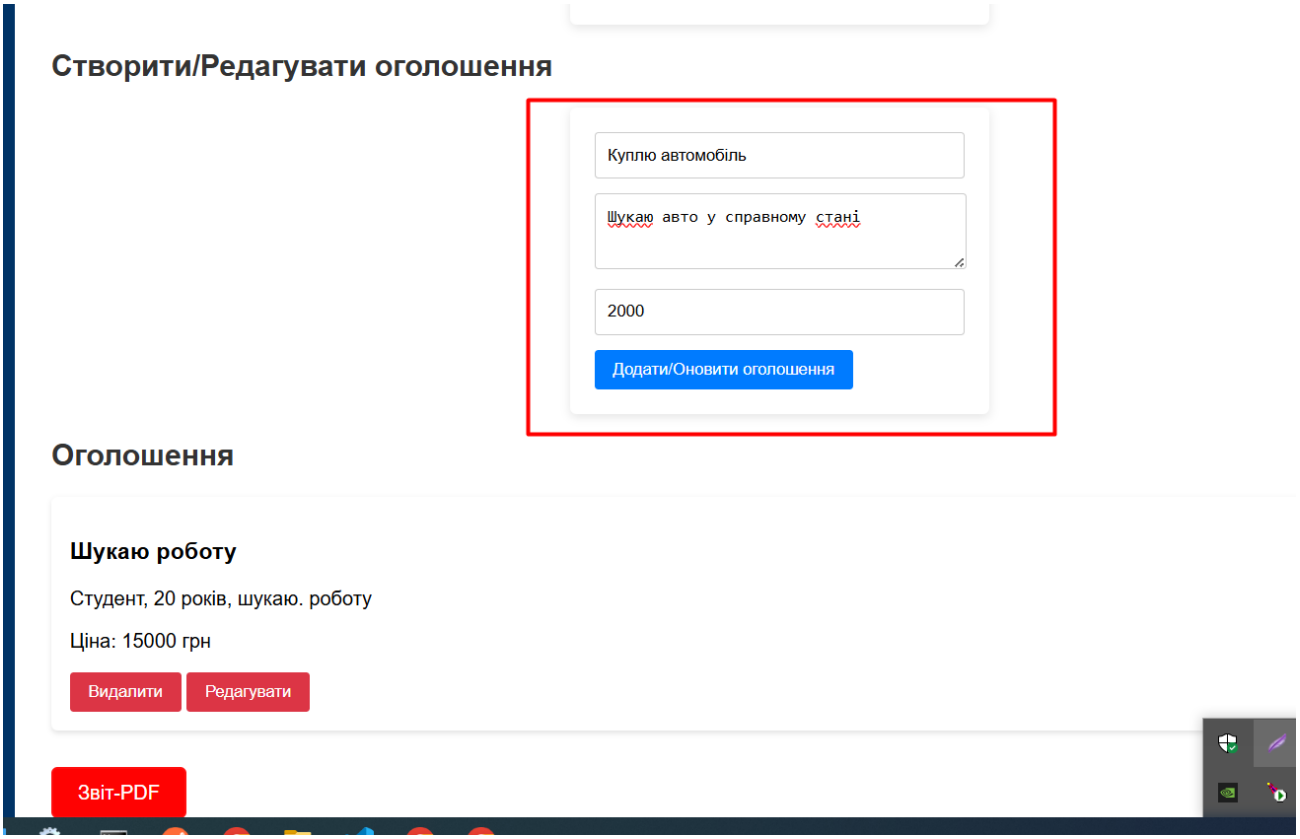
Вхід

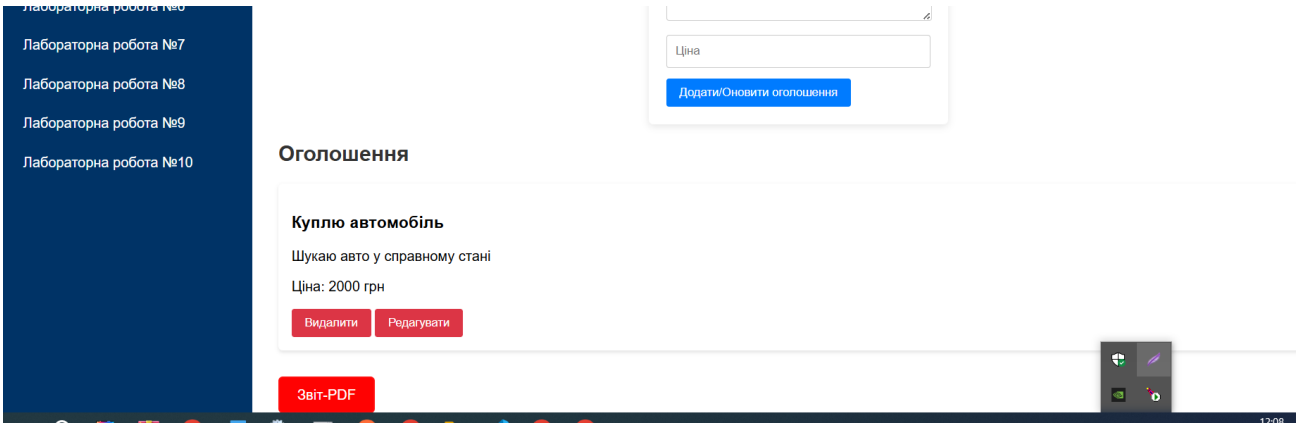


Створення оголошення

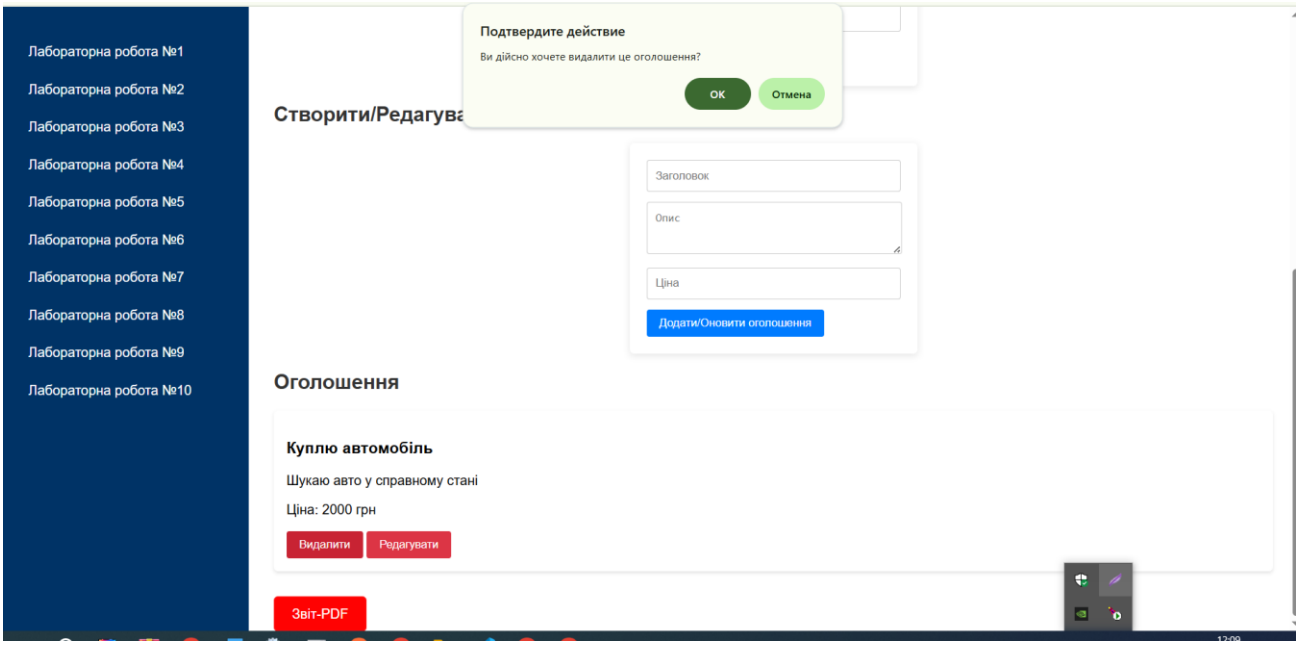


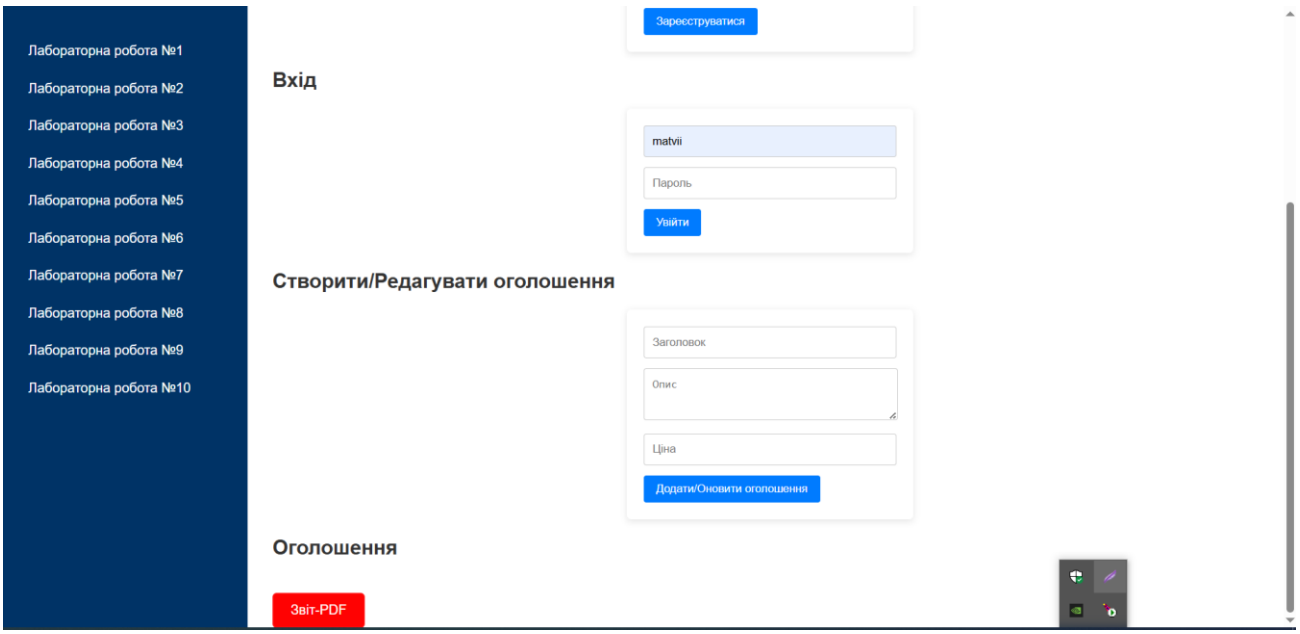
Зміна оголошення





Видалення оголошення





Коди:

Server_lab.js

```
const express = require("express");
const http = require("http");
const socketIo = require("socket.io");
const cors = require("cors");
const jwt = require("jsonwebtoken");
const bodyParser = require("body-parser");
const swaggerUi = require("swagger-ui-express");
```

```
const fs = require("fs");

const app = express();
const server = http.createServer(app);
const io = socketIo(server, { cors: { origin: "*" } });

const PORT = 5000;
const SECRET_KEY = "my_secret_key";

app.use(cors());
app.use(bodyParser.json());
app.use(express.static("public"));

// Swagger setup
const swaggerDocument = JSON.parse(fs.readFileSync("./swagger.json", "utf8"));
app.use("/api-docs", swaggerUi.serve, swaggerUi.setup(swaggerDocument));

let users = []; // Temporary in-memory storage
let ads = []; // In-memory ads
let nextAdId = 1; // To generate unique ad IDs

// Register endpoint
app.post("/api/register", (req, res) => {
    const { username, password } = req.body;
    if (users.find(u => u.username === username)) {
        return res.status(400).json({ message: "Користувач вже існує." });
    }
    users.push({ username, password });
    res.status(201).json({ message: "Користувач зареєстрований успішно!" });
});

// Login endpoint
app.post("/api/login", (req, res) => {
    const { username, password } = req.body;
    const user = users.find(u => u.username === username && u.password ===
password);
    if (!user) {
        return res.status(401).json({ message: "Невірний логін або пароль." });
    }
    const token = jwt.sign({ username }, SECRET_KEY);
    res.json({ token });
});

// Middleware to authenticate JWT
function authenticateToken(req, res, next) {
    const authHeader = req.headers["authorization"];
    const token = authHeader && authHeader.split(" ")[1];
```

```
    if (!token) return res.sendStatus(401);

    jwt.verify(token, SECRET_KEY, (err, user) => {
        if (err) return res.sendStatus(403);
        req.user = user;
        next();
    });
}

// Get all ads
app.get("/api/ads", authenticateToken, (req, res) => {
    res.json(ads);
});

// Post a new ad with duplicate check
app.post("/api/ads", authenticateToken, (req, res) => {
    const { title, description, price } = req.body;

    const existingAd = ads.find(ad => ad.title === title && ad.description ===
description);
    if (existingAd) {
        return res.status(400).json({ message: "Оголошення з таким заголовком і
описом вже існує" });
    }

    const ad = {
        id: nextAdId++,
        title,
        description,
        price,
        username: req.user.username,
    };

    ads.push(ad);

    // io.emit("newAd", ad); // Убираємо emit для уникнення дублювання

    res.status(201).json(ad);
});

// Update an ad
app.put("/api/ads/:id", authenticateToken, (req, res) => {
    const adIndex = ads.findIndex(ad => ad.id === parseInt(req.params.id));
    if (adIndex === -1) return res.status(404).json({ message: "Оголошення не
знайдено" });

    // Check if the user is the owner of the ad
```

```
    if (ads[adIndex].username !== req.user.username) {
        return res.status(403).json({ message: "Неможливо редагувати це оголошення"
});
    }

    // Update the ad
    const updatedAd = { ...ads[adIndex], ...req.body };
    ads[adIndex] = updatedAd;
    res.status(200).json(updatedAd);
});

// Delete an ad
app.delete("/api/ads/:id", authenticateToken, (req, res) => {
    const adIndex = ads.findIndex(ad => ad.id === parseInt(req.params.id));
    if (adIndex === -1) return res.status(404).json({ message: "Оголошення не
знайдено" });

    // Check if the user is the owner of the ad
    if (ads[adIndex].username !== req.user.username) {
        return res.status(403).json({ message: "Неможливо видалити це оголошення"
});
    }

    // Delete the ad
    ads.splice(adIndex, 1);
    res.status(200).json({ message: "Оголошення видалено" });
});

// Socket.io connection
io.on("connection", (socket) => {
    console.log("Socket connected: " + socket.id);
});

server.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
    console.log(`Swagger docs: http://localhost:${PORT}/api-docs`);
});
```

Ad.Controler

```
const Ad = require("../models/Ad");
const jwt = require("jsonwebtoken");

exports.getAds = async (req, res) => {
  const ads = await Ad.find();
  res.json(ads);
```

```
};

exports.createAd = async (req, res) => {
  const { title, description, price } = req.body;
  const username = req.user.username;

  const ad = new Ad({ title, description, price, username });
  await ad.save();

  req.io.emit("newAd", ad);
  res.status(201).json(ad);
};

exports.updateAd = async (req, res) => {
  const adId = req.params.id;
  const username = req.user.username;

  const ad = await Ad.findById(adId);
  if (!ad) return res.status(404).json({ message: "Оголошення не знайдено" });

  if (ad.username !== username) {
    return res.status(403).json({ message: "Ви не можете редагувати це оголошення"
});
  }

  ad.title = req.body.title;
  ad.description = req.body.description;
  ad.price = req.body.price;
  await ad.save();

  req.io.emit("updateAd", ad);
  res.json(ad);
};

exports.deleteAd = async (req, res) => {
  const adId = req.params.id;
  const username = req.user.username;

  const ad = await Ad.findById(adId);
  if (!ad) return res.status(404).json({ message: "Оголошення не знайдено" });

  if (ad.username !== username) {
    return res.status(403).json({ message: "Ви не можете видалити це оголошення"
});
  }

  await ad.deleteOne();
```

```
  req.io.emit("deleteAd", adId);
  res.json({ message: "Оголошення видалено" });
};
```

authMiddleware

```
const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {
  const token = req.header("Authorization");
  if (!token) return res.status(401).json({ message: "Отсутствует токен" });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded.user;
    next();
  } catch (error) {
    res.status(401).json({ message: "Неверный токен" });
  }
};

module.exports = authMiddleware;
```

authentication

```
const express = require("express");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");
const router = express.Router();

const users = [];

router.post("/register", async (req, res) => {
  const { username, password } = req.body;
  const existingUser = users.find(user => user.username === username);
  if (existingUser) {
    return res.status(400).json({ message: "Користувач вже існує" });
  }
  const hashedPassword = await bcrypt.hash(password, 10);
  users.push({ username, password: hashedPassword });
  res.status(201).json({ message: "Реєстрація успішна" });
});
```

```
router.post("/login", async (req, res) => {
  const { username, password } = req.body;
  const user = users.find(user => user.username === username);
  if (!user) {
    return res.status(400).json({ message: "Користувач не знайдений" });
  }
  const isPasswordValid = await bcrypt.compare(password, user.password);
  if (!isPasswordValid) {
    return res.status(400).json({ message: "Невірний пароль" });
  }
  const token = jwt.sign({ username }, "secretkey", { expiresIn: "1h" });
  res.json({ token });
});

module.exports = router;
```

control_ad.js

```
const express = require("express");
const router = express.Router();
const adController = require("../controllers/adController");
const jwt = require("jsonwebtoken");

function authenticateToken(req, res, next) {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  if (!token) return res.sendStatus(401);

  jwt.verify(token, process.env.JWT_SECRET || "secretkey", (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

router.get("/", authenticateToken, adController.getAds);
router.post("/", authenticateToken, adController.createAd);
router.put("/:id", authenticateToken, adController.updateAd);
router.delete("/:id", authenticateToken, adController.deleteAd);

module.exports = router;
```

lab5.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Лабораторна робота №5</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <style>



.pdf-button
{

    margin-top: 20px;
    display: inline-block;
    padding: 10px 20px;
    font-size: 16px;
    color: #fff;
    background-color: #ff0202;
    text-decoration: none;
    border-radius: 5px;
    border: 2px solid #ff0f0f;
    transition: background-color 0.3s, transform 0.2s;

}


.pdf-button:hover {
    background-color: #fe3213;
    transform: scale(1.05);
}

.pdf-button:active {
    background-color: #ff2222;
}



p
{
```

```
    width: 900px
}

/*NEW*/

h2 {
            color: #333;
        }
        form {
    width: 300px; /* фиксированная ширина */
    margin: 20px auto; /* центрируем форму */
    background: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

input, textarea {
    width: 100%; /* поля на всю ширину формы */
    margin-bottom: 12px;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 3px;
    box-sizing: border-box; /* чтобы padding не ломал ширину */
}

        button {
            padding: 8px 15px;
            background-color: #007bff;
            border: none;
            color: white;
            cursor: pointer;
            border-radius: 3px;
        }
        button:hover {
            background-color: #0056b3;
        }
        .ad {
            background: #fff;
            padding: 15px;
            margin-bottom: 10px;
            border-radius: 5px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }
        .ad button {
            background-color: #dc3545;
        }
```

```
        .ad button:hover {
            background-color: #c82333;
        }


    </style>




    <div class="container">
        <!-- Бічна панель -->
        <nav class="sidebar">
            <ul>
                <li><a href="lab1.html">Лабораторна робота №1</a></li>
                <li><a href="lab2.html">Лабораторна робота №2</a></li>
                <li><a href="lab3.html">Лабораторна робота №3</a></li>
                <li><a href="lab4.html">Лабораторна робота №4</a></li>
                <li><a href="lab5-копия.html">Лабораторна робота №5</a></li>
                <li><a href="lab6.html">Лабораторна робота №6</a></li>
                <li><a href="lab7.html">Лабораторна робота №7</a></li>
                <li><a href="lab8.html">Лабораторна робота №8</a></li>
                <li><a href="lab9.html">Лабораторна робота №9</a></li>
                <li><a href="lab10.html">Лабораторна робота №10</a></li>
            </ul>
        </nav>

        <!-- Основний контент -->
        <main class="content">
            <h1>Лабораторна робота №5</h1>
            <div class="buttons">
                <button class="custom-button"
onclick="toggleContent('condition')">Умова</button>
                <button class="custom-button"
onclick="toggleContent('result')">Результат</button>
                <button class="custom-button"
onclick="toggleContent('content')">Код</button>

            </div>
            <div id="condition" class="hidden-content">
                <h3>
                    20. Створити API для дошки оголошень: CRUD для оголошень,
авторизація JWT,
                    WebSockets для миттєвого оновлення нових пропозицій.

                </h3>
```

```
            </div>
            <div id="result" class="hidden-content">

                <h2>Реєстрація</h2>
                <form id="registerForm">
                    <input type="text" id="registerUsername" placeholder="Ім'я
користувача" required />
                    <input type="password" id="registerPassword"
placeholder="Пароль" required />
                    <button type="submit">Зареєструватися</button>
                </form>

                <h2>Вхід</h2>
                <form id="loginForm">
                    <input type="text" id="loginUsername" placeholder="Ім'я
користувача" required />
                    <input type="password" id="loginPassword" placeholder="Пароль"
required />
                    <button type="submit">Увійти</button>
                </form>

                <h2>Створити/Редагувати оголошення</h2>
                <form id="adForm" style="display:none;">
                    <input type="text" id="adTitle" placeholder="Заголовок"
required />
                    <textarea id="adDescription" placeholder="Опис"
required></textarea>
                    <input type="number" id="adPrice" placeholder="Ціна" required
/>
                    <button type="submit">Додати/Оновити оголошення</button>
                </form>

                <h2>Оголошення</h2>
                <div id="adsContainer"></div>

                <script>
                    let token = null;
                    let ads = [];
                    let editingAdId = null;

                    document.getElementById("registerForm").addEventListener("submi
t", async (e) => {
                        e.preventDefault();
                        const username =
document.getElementById("registerUsername").value;
                        const password =
document.getElementById("registerPassword").value;
```

```
                        try {
                            const res = await
fetch("http://localhost:5000/api/register", {
                                method: "POST",
                                headers: { "Content-Type": "application/json" },
                                body: JSON.stringify({ username, password }),
                            });

                            const data = await res.json();
                            alert(data.message);
                        } catch (error) {
                            alert("Помилка при реєстрації");
                        }
                    });

                    document.getElementById("loginForm").addEventListener("submit",
async (e) => {
                        e.preventDefault();
                        const username =
document.getElementById("loginUsername").value;
                        const password =
document.getElementById("loginPassword").value;

                        try {
                            const res = await
fetch("http://localhost:5000/api/login", {
                                method: "POST",
                                headers: { "Content-Type": "application/json" },
                                body: JSON.stringify({ username, password }),
                            });

                            const data = await res.json();
                            if (res.ok) {
                                token = data.token;
                                alert("Успішний вхід!");
                                document.getElementById("adForm").style.display =
"block";
                                loadAds();
                            } else {
                                alert(data.message);
                            }
                        } catch (error) {
                            alert("Помилка при вході");
                        }
                    });
```

```
                    document.getElementById("adForm").addEventListener("submit",
async (e) => {
                        e.preventDefault();
                        const title = document.getElementById("adTitle").value;
                        const description =
document.getElementById("adDescription").value;
                        const price = document.getElementById("adPrice").value;

                        const method = editingAdId ? "PUT" : "POST";
                        const url = editingAdId ?
`http://localhost:5000/api/ads/${editingAdId}` : "http://localhost:5000/api/ads";

                        try {
                            const res = await fetch(url, {
                                method: method,
                                headers: {
                                    "Content-Type": "application/json",
                                    "Authorization": "Bearer " + token,
                                },
                                body: JSON.stringify({ title, description, price
}),
                            });

                            const ad = await res.json();
                            if (res.ok) {
                                document.getElementById("adForm").reset();
                                editingAdId = null;  // Reset the editing state
                                loadAds();  // Перезавантажити оголошення
                            } else {
                                alert(ad.message || "Помилка при
створенні/оновленні оголошення");
                            }
                        } catch (error) {
                            alert("Помилка при створенні/оновленні оголошення");
                        }
                    });

                    async function loadAds() {
                        if (!token) {
                            alert("Для перегляду оголошень потрібно увійти");
                            return;
                        }
                        try {
                            const res = await
fetch("http://localhost:5000/api/ads", {
                                headers: { "Authorization": "Bearer " + token }
                            });
```

```
                        const data = await res.json();
                        ads = data;
                        renderAds();
                    } catch (error) {
                        alert("Помилка при завантаженні оголошень");
                    }
                }

                function renderAds() {
                    const adsContainer =
document.getElementById("adsContainer");
                    adsContainer.innerHTML = "";
                    ads.forEach(ad => {
                        const adElement = document.createElement("div");
                        adElement.classList.add("ad");
                        adElement.innerHTML = `
                            <h3>${ad.title}</h3>
                            <p>${ad.description}</p>
                            <p>Ціна: ${ad.price} грн</p>
                            <button
onclick="deleteAd(${ad.id})">Видалити</button>
                            <button
onclick="editAd(${ad.id})">Редагувати</button>
                        `;
                        adsContainer.appendChild(adElement);
                    });
                }

                async function deleteAd(id) {
                    const confirmation = confirm("Ви дійсно хочете видалити це
оголошення?");
                    if (!confirmation) return;

                    try {
                        const res = await
fetch(`http://localhost:5000/api/ads/${id}`, {
                            method: "DELETE",
                            headers: { "Authorization": "Bearer " + token }
                        });

                        const data = await res.json();
                        if (res.ok) {
                            loadAds();
                        } else {
                            alert(data.message);
                        }
                    } catch (error) {
```

```
                            alert("Помилка при видаленні оголошення");
                        }
                    }

                    function editAd(id) {
                        const ad = ads.find(ad => ad.id === id);
                        if (!ad) return alert("Оголошення не знайдено");

                        document.getElementById("adTitle").value = ad.title;
                        document.getElementById("adDescription").value =
ad.description;
                        document.getElementById("adPrice").value = ad.price;

                        editingAdId = ad.id;  // Set the ID for editing
                    }
                </script>

            <a href="лаб-2.1.pdf" class="pdf-button"  target="_blank">Звіт-PDF</a>


            </div>
            <div id="content" class="hidden-content">

                <a href="https://github.com/YuShChEnKMaTvIi/STIP-2_LABS_CODES"
class="github-button" target="_blank">Посилання на GitHub</a>

            </div>
        </main>
    </div>
    <script src="scripts.js"></script>
</body>
</html>
```

Swagger.json

```
{
    "openapi": "3.0.0",
    "info": {
      "title": "Дошка оголошень API",
      "version": "1.0.0"
    },
    "paths": {
      "/api/register": {
        "post": {
          "summary": "Реєстрація користувача",
```

```
      "requestBody": {
        "required": true,
        "content": {
          "application/json": {
            "schema": {
              "type": "object",
              "properties": {
                "username": { "type": "string" },
                "password": { "type": "string" }
              },
              "required": ["username", "password"]
            }
          }
        }
      },
      "responses": {
        "201": { "description": "Успішна реєстрація" },
        "400": { "description": "Користувач вже існує" }
      }
    }
  },
  "/api/login": {
    "post": {
      "summary": "Логін користувача",
      "requestBody": {
        "required": true,
        "content": {
          "application/json": {
            "schema": {
              "type": "object",
              "properties": {
                "username": { "type": "string" },
                "password": { "type": "string" }
              },
              "required": ["username", "password"]
            }
          }
        }
      },
      "responses": {
        "200": { "description": "Успішний логін" },
        "401": { "description": "Невірний логін або пароль" }
      }
    }
  },
  "/api/ads": {
    "get": {
```

```
        "summary": "Отримати всі оголошення",
        "security": [{ "bearerAuth": [] }],
        "responses": {
          "200": { "description": "Список оголошень" },
          "401": { "description": "Неавторизовано" }
        }
      },
      "post": {
        "summary": "Створити нове оголошення",
        "security": [{ "bearerAuth": [] }],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "type": "object",
                "properties": {
                  "title": { "type": "string" },
                  "description": { "type": "string" },
                  "price": { "type": "number" }
                },
                "required": ["title", "description", "price"]
              }
            }
          }
        },
        "responses": {
          "201": { "description": "Оголошення створено" },
          "401": { "description": "Неавторизовано" }
        }
      }
    },
    "components": {
      "securitySchemes": {
        "bearerAuth": {
          "type": "http",
          "scheme": "bearer",
          "bearerFormat": "JWT"
        }
      }
    }
  }
```