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 лабораторної роботи №7 з дисципліни
«Програмування веб-застосувань»

«Експерименти із підвищеним навантаженням»
Варіант 13

Виконав *ІП-33 Хребтань Максим Віталійович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Очеретяний Олександр Костянтинович*
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №7

Експерименти із підвищеним навантаженням

Мета роботи: Ознайомлення із роботою серверів та скриптів в режимі підвищеного навантаження.

Завдання:

1. Оберіть сервер (хостінг або сервіс розгортання веб-додатків) для розміщення функціонального прикладу виконаного завдання.
2. За допомогою HTML, CSS та мов програмування Javascript і PHP виконайте маніпуляції з веб-сторінками згідно з вашим варіантом:
    a. Створіть копію веб-сторінки із 3 л.р., в ній за допомогою HTML, CSS та JS розробіть код для функціонування об'єкта згідно із вашим варіантом завдання. Сформований об'єкт повинен функціонувати без використання JS та CSS-фреймворків (JQuery, Bootstrap та ін.).
    b. Створіть перший спосіб зберігання на сервері тих подій, які відбуваються в об'єкті, через негайне відправляння на сервер даних кожної події об'єкта, із порядковим номером події та зафіксованим часом збереження даних на сервері. Порівняйте збережений серверний час із локальним часом вашого ПК, за необхідності внесіть правки у ваш PHP-код фіксування часу на сервері для уникнення суттєвих (в кілька годин) розбіжностей.
    c. Створіть другий спосіб зберігання на сервері тих подій, які відбуваються в об'єкті, через одне кінцеве відправляння на сервер акумульованих в LocalStorage даних кожної події об'єкта, із порядковим номером події та зафіксованим часом збереження даних в LocalStorage. Акумуляція даних в LocalStorage повинна відбуватися поруч (практично одночасно) із негайним відправлянням даних на сервер із першого способу.
    d. Порівняйте початкові та кінцеві часи збережених обома способами даних, порівняйте різниці в часах між різними подіями для обох способів, виявлені розбіжності або відсутність розбіжностей у збережених даних опишіть у звіті. Якщо помітили нестандартну поведінку веб-сторінки – згадайте про це у звіті.
3. Для запису стилів і скриптів використовуйте будь-які відомі вам способи.
4. Файли виконаного завдання розмістіть в репозиторії на Github, функціональний приклад виконаного завдання розмістіть на сервері та продемонструйте його роботу або зафіксуйте його роботу на відео

Варіант 13

a) Робоче поле для анімації розміщується у
окремому шарі «work» розміром (W(45%)+10px) x
(H(85%)+50px) поверх веб-сторінки.

b) Область анімації «anim» розміщена всередині
робочого поля «work» на площі (W(45%) x H(85%)),
із зовнішньою червоною рамкою 5px, притиснута до
нижнього краю «work».

c) Решта «work» з умовною назвою «controls»
призначена для написів і керуючих елементів.

d) «work» появляється при натисканні кнопки «play»
у блоці «5» і зникає при натисканні кнопки «close», розміщеної у «controls» зліва.

e) Верхня і нижня половини фону «anim» заповнюються двома різними текстурами розміром 32px x 32px, підготовленими вами у графічному редакторі (аргументуйте вибір редактора і формату).

f) Два квадрати зі стороною 15px зеленого і оранжевого кольору появляються біля верхньої і нижньої стінок «anim» з випадковою горизонтальною координатою, рухаються під випадковими кутами з однією швидкістю до протилежних стінок, дотик до стінок змінює напрямок згідно з ідеальною фізичною моделлю, дотик квадратів між собою змінює їхні напрямки на протилежні; повне розміщення обох квадратів на верхній чи нижній половині «anim» зупиняє анімацію.

g) Квадрати починають рух при натисканні кнопки «start», після чого «start» втрачає можливість натискання. Якщо квадрати опинились на верхній/нижній половині «anim», на місці кнопки «start» появляється кнопка «reload», яка дозволяє поставити квадрати на нові місця і розмістити замість себе «start».

h) Натискання кнопок, дотики квадратів до стінок і між собою, кожен крок/зміщення квадратів, повне розміщення обох квадратів на верхній чи нижній половині «anim» є причинами появи текстового повідомлення справа у «controls», і є подіями для зберігання двома способами на сервері із вказанням порядкового номеру події, часу, та вищезгаданого текстового повідомлення. Після натискання кнопки «close» всі ці записи зчитуються з LocalStorage та сервера, і показуються у блоці «4» у двох колонках таблиці.

i) Часовий проміжок між кроками/зміщеннями круга підберіть достатньо невеликий для швидкого відпрацювання анімації.

ХІД РОБОТИ

Робоче поле «work»:

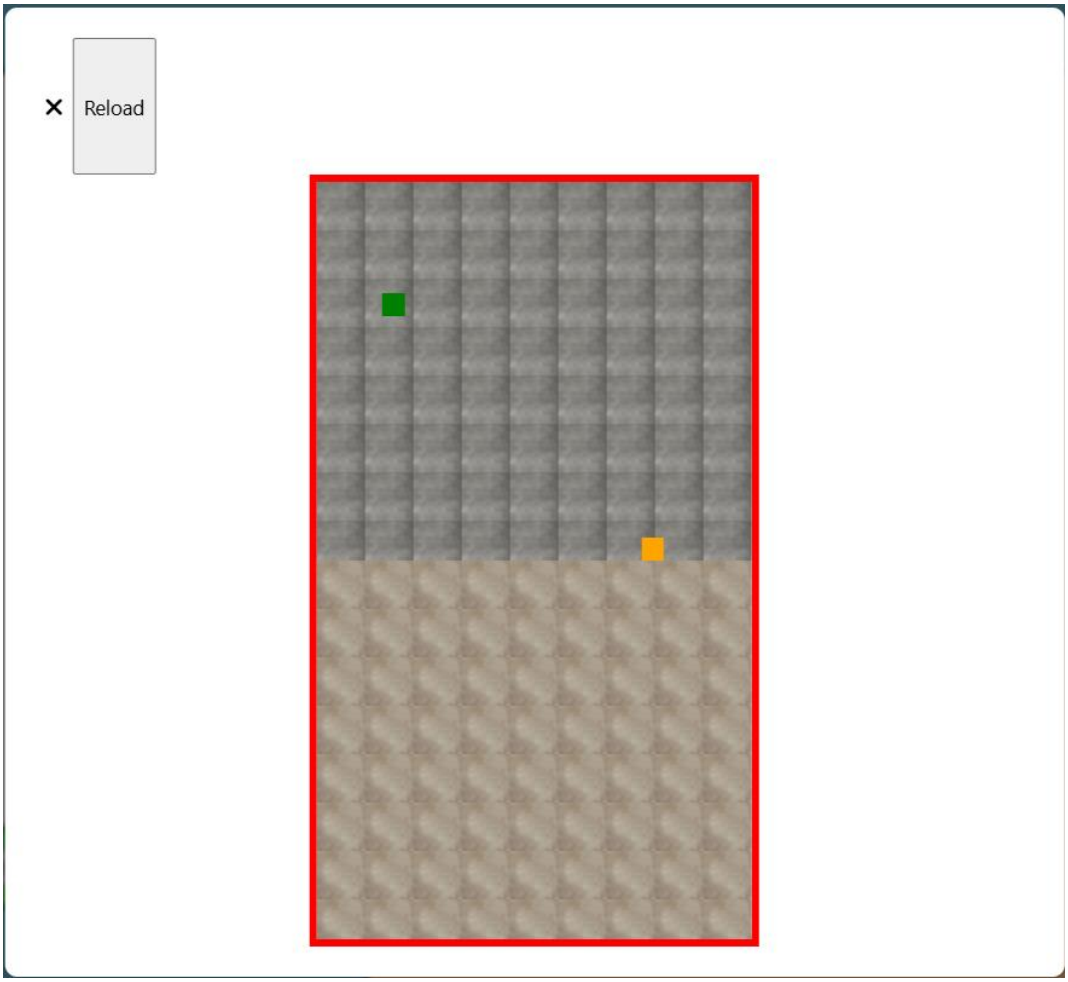

Рис. 1. Створений елемент

Таблиця подій:

Метод 1	Метод 2
Client: 2024-12-15T15:36:12.260Z | Server: 2024-12-15 17:36:14 - 1.Start	Client: 2024-12-15T15:36:12.261Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 1.Start
Client: 2024-12-15T15:36:12.267Z | Server: 2024-12-15 17:36:14 - 2.green square move	Client: 2024-12-15T15:36:12.267Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 2.green square move
Client: 2024-12-15T15:36:12.268Z | Server: 2024-12-15 17:36:14 - 3.orange square move	Client: 2024-12-15T15:36:12.268Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 3.orange square move
Client: 2024-12-15T15:36:12.272Z | Server: 2024-12-15 17:36:14 - 4.green square move	Client: 2024-12-15T15:36:12.273Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 4.green square move
Client: 2024-12-15T15:36:12.273Z | Server: 2024-12-15 17:36:14 - 5.orange square move	Client: 2024-12-15T15:36:12.273Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 5.orange square move
Client: 2024-12-15T15:36:12.278Z | Server: 2024-12-15 17:36:14 - 6.green square move	Client: 2024-12-15T15:36:12.278Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 6.green square move
Client: 2024-12-15T15:36:12.283Z | Server: 2024-12-15 17:36:14 - 8.green square move	Client: 2024-12-15T15:36:12.279Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 7.orange square move
Client: 2024-12-15T15:36:12.284Z | Server: 2024-12-15 17:36:14 - 9.orange square move	Client: 2024-12-15T15:36:12.284Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 8.green square move
Client: 2024-12-15T15:36:12.287Z | Server: 2024-12-15 17:36:14 - 11.orange square move	Client: 2024-12-15T15:36:12.284Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 9.orange square move
Client: 2024-12-15T15:36:12.292Z | Server: 2024-12-15 17:36:14 - 12.green square move	Client: 2024-12-15T15:36:12.287Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 10.green square move
Client: 2024-12-15T15:36:12.279Z | Server: 2024-12-15 17:36:14 - 7.orange square move	Client: 2024-12-15T15:36:12.288Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 11.orange square move
Client: 2024-12-15T15:36:12.287Z | Server: 2024-12-15 17:36:15 - 10.green square move	Client: 2024-12-15T15:36:12.292Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 12.green square move
Client: 2024-12-15T15:36:12.293Z | Server: 2024-12-15 17:36:15 - 13.orange square move	Client: 2024-12-15T15:36:12.293Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15 17:37:21 - 13.orange square move
Client: 2024-12-15T15:36:12.298Z | Server: 2024-12-15 17:36:15 -	Client: 2024-12-15T15:36:12.298Z | Sent: 2024-12-15T15:37:19.371Z | Server: 2024-12-15

Рис. 2. Таблиця із отриманими даними з серверу

Код

Script.js

```
document.addEventListener("DOMContentLoaded", function() {
    const button = document.getElementById("openPopup");
    const popup = document.getElementById("popup");
    const overlay = document.getElementById("blurOverlay");
    const buttonClose = document.getElementById("closePopup");

    function openPopup() {
        localStorage.clear();
        deleteLogs();
        inTimeEvents = 1;
        localEvents = 1;
        popup.classList.add("active");
        overlay.classList.add("active");
    }

    function closePopup() {
        sendLocalEventsToServer();
        fetchLogs();
        popup.classList.remove("active");
        overlay.classList.remove("active");
    }

    button.addEventListener("click", openPopup);

    buttonClose.addEventListener("click", closePopup);
});

let interval;
let squares = [];

function createSquare(color, container, isTop) {
    const square = document.createElement("div");
    square.classList.add("square", color);

    const containerWidth = container.clientWidth;
    const containerHeight = container.clientHeight;
    const squareSize = 15;

    const x = Math.random() * (containerWidth - squareSize);
    const y = isTop ? 0 : containerHeight - squareSize;

    square.style.left = `${x}px`;
    square.style.top = `${y}px`;

    const angle = Math.random() * 2 * Math.PI;
    square.direction = {
        dx: Math.cos(angle),
        dy: Math.sin(angle),
    };

    square.speed = 2;
    square.size = squareSize;
    square.color = color;
```

```
        container.appendChild(square);
        return square;
    }

    function initSquares() {
        const animContainer = document.getElementById("anim");
        animContainer.innerHTML = "";

        squares = [
            createSquare("green", animContainer, true),
            createSquare("orange", animContainer, false),
        ];
    }

    function moveSquares() {
        const animContainer = document.getElementById("anim");
        const containerWidth = animContainer.clientWidth;
        const containerHeight = animContainer.clientHeight;

        let squaresInTopHalf = 0;
        let squaresInBottomHalf = 0;

        squares.forEach((square) => {
            let x = parseFloat(square.style.left) + square.direction.dx * square.speed;
            let y = parseFloat(square.style.top) + square.direction.dy * square.speed;

            square.style.left = `${Math.min(Math.max(0, x), containerWidth -
square.size)}px`;
            square.style.top = `${Math.min(Math.max(0, y), containerHeight -
square.size)}px`;
            sendEventToServer(`${square.color} square move`);
            saveEventToLocal(`${square.color} square move`);

            if (x <= 0 || x + square.size >= containerWidth) {
                square.direction.dx *= -1;
                sendEventToServer(`${square.color} square collided with the wall`);
                saveEventToLocal(`${square.color} square collided with the wall`);
            }
            if (y <= 0 || y + square.size >= containerHeight) {
                square.direction.dy *= -1;
                sendEventToServer(`${square.color} square collided with the wall`);
                saveEventToLocal(`${square.color} square collided with the wall`);
            }

            if (y + square.size <= containerHeight / 2) {
                squaresInTopHalf++;
            } else if (y >= containerHeight / 2) {
                squaresInBottomHalf++;
            }
        });

        const dx = parseFloat(squares[0].style.left) - parseFloat(squares[1].style.left);
        const dy = parseFloat(squares[0].style.top) - parseFloat(squares[1].style.top);
        const distance = Math.sqrt(dx * dx + dy * dy);
        if (distance < squares[0].size) {
            squares.forEach((square) => {
```

```
            square.direction.dx *= -1;
            square.direction.dy *= -1;
        });
        sendEventToServer("Squares collided");
        saveEventToLocal("Squares collided");
    }

    if (squaresInTopHalf === 2 || squaresInBottomHalf === 2) {
        sendEventToServer("In one half");
        saveEventToLocal("In one half");
        clearInterval(interval);
        document.getElementById("start").style.display = "none";
        document.getElementById("reload").style.display = "inline-block";
    }
}

function startBtnFunc() {
    initSquares();
    sendEventToServer("Start");
    saveEventToLocal("Start");
    interval = setInterval(moveSquares, 5);

    document.getElementById("start").disabled = true;
    document.getElementById("reload").style.display = "none";
}

function reloadBtnFunc() {
    clearInterval(interval);
    initSquares();
    sendEventToServer("Reload");
    saveEventToLocal("Reload");

    document.getElementById("start").disabled = false;
    document.getElementById("start").style.display = "inline-block";
    document.getElementById("reload").style.display = "none";
}

let inTimeEvents = 1;

function sendEventToServer(eventName) {
    const now = new Date();
    const timestamp = now.toISOString();

    const data = {
        event: `${inTimeEvents}.${eventName}`,
        clientTime: timestamp
    };

    inTimeEvents++;

    fetch('method1.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(data),
    })
```

```
    .then(response => response.json())
    .then(result => {
        if (result.status === 'success') {
            console.log(`Подія "${eventName}" успішно збережена. Час: ${timestamp}`);
        } else {
            console.error('Помилка збереження події:', result.message);
        }
    })
    .catch(error => console.error('Помилка запиту:', error));
}

let localEvents = 1;

function saveEventToLocal(eventName) {
    const now = new Date();
    const timestamp = now.toISOString();

    const event = {
        event: `${localEvents}.${eventName}`,
        clientTime: timestamp
    };

    localEvents++;

    let events = JSON.parse(localStorage.getItem("events")) || [];

    events.push(event);

    localStorage.setItem("events", JSON.stringify(events));

    console.log(`Подія "${eventName}" збережена в localStorage. Час: ${timestamp}`);
}

function sendLocalEventsToServer() {
    const events = JSON.parse(localStorage.getItem("events")) || [];

    if (events.length === 0) {
        console.log("Немає подій для відправки.");
        return;
    }

    const now = new Date();
    const sendTime = now.toISOString();

    const data = {
        events: events,
        sendTime: sendTime
    };

    fetch('method2.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(data),
    })
    .then(response => response.json())
```

```
    .then(result => {
        if (result.status === 'success') {
            console.log(`Події успішно надіслані. Час відправлення: ${sendTime}`);

            localStorage.removeItem("events");
        } else {
            console.error('Помилка збереження подій:', result.message);
        }
    })
    .catch(error => console.error('Помилка запиту:', error));
}

function fetchLogs() {
    fetch('get_logs.php')
        .then(response => response.json())
        .then(data => {
            const method1Logs = data.method1;
            const method2Logs = data.method2;

            createTable(method1Logs, method2Logs);
        })
        .catch(error => console.error('Помилка завантаження логів:', error));
}

function createTable(method1Logs, method2Logs) {
    const table = document.createElement('table');
    table.border = '1';
    table.style.width = '100%';

    const headerRow = document.createElement('tr');
    const method1Header = document.createElement('th');
    method1Header.textContent = 'Метод 1';
    const method2Header = document.createElement('th');
    method2Header.textContent = 'Метод 2';

    headerRow.appendChild(method1Header);
    headerRow.appendChild(method2Header);
    table.appendChild(headerRow);

    const maxRows = Math.max(method1Logs.length, method2Logs.length);

    for (let i = 0; i < maxRows; i++) {
        const row = document.createElement('tr');


        const method1Cell = document.createElement('td');
        method1Cell.textContent = method1Logs[i] || '';
        row.appendChild(method1Cell);


        const method2Cell = document.createElement('td');
        method2Cell.textContent = method2Logs[i] || '';
        row.appendChild(method2Cell);

        table.appendChild(row);
    }
```

```
    const container = document.getElementById('logTableContainer');
    container.innerHTML = '';
    container.appendChild(table);
}

window.addEventListener('unload', clear);

function clear() {
    localStorage.clear();
    deleteLogs();
}

async function deleteLogs() {
    try {
        const response = await fetch('delete_logs.php', { method: "DELETE" });

        if (response.ok) {
            console.log("Логи успішно видалені!");
        } else {
            const error = await response.text();
            console.log("Помилка при видаленні логів:", error);
        }
    } catch (err) {
        console.error("Помилка при видаленні логів:", err);
    }
}
```

Method1.php

```
<?php
date_default_timezone_set('Europe/Kiev');

$filename = 'events_log1.txt';

$data = json_decode(file_get_contents('php://input'), true);

if ($data) {
    $event = $data['event'];
    $clientTime = $data['clientTime'];
    $serverTime = date('Y-m-d H:i:s');

    $logLine = "Client: $clientTime | Server: $serverTime - $event\n";

    file_put_contents($filename, $logLine, FILE_APPEND | LOCK_EX);

    echo json_encode(['status' => 'success', 'message' => 'Event saved!']);
} else {
    http_response_code(400);
    echo json_encode(['status' => 'error', 'message' => 'No data received']);
}
?>
```

Method2.php

```
<?php
date_default_timezone_set('Europe/Kiev');
```

```
$filename = 'events_log2.txt';

$data = json_decode(file_get_contents('php://input'), true);

if ($data) {
    $events = $data['events'];
    $sendTime = $data['sendTime'];

    foreach ($events as $event) {
        $serverTime = date('Y-m-d H:i:s');
        $logLines .= "Client: {$event['clientTime']} | Sent: {$sendTime} | Server:
{$serverTime} - {$event['event']}\n";
    }

    file_put_contents($filename, $logLines, FILE_APPEND | LOCK_EX);

    echo json_encode(['status' => 'success', 'message' => 'Events saved!']);
} else {
    http_response_code(400);
    echo json_encode(['status' => 'error', 'message' => 'No data received']);
}
?>
```

Get_logs.php

```
<?php
$method1File = 'events_log1.txt';
$method2File = 'events_log2.txt';

function getLogs($filename) {
    if (file_exists($filename)) {
        $lines = file($filename, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
        $logs = [];

        foreach ($lines as $line) {
            $logs[] = trim($line);
        }
        return $logs;
    }
    return [];
}

$method1Logs = getLogs($method1File);
$method2Logs = getLogs($method2File);

echo json_encode([
    'method1' => $method1Logs,
    'method2' => $method2Logs
]);
?>
```

Посилання

–Репозиторій

–Веб сторінка

ВИСНОВОК

У ході виконання лабораторної роботи було досліджено принципи роботи серверів і скриптів в умовах підвищеного навантаження. Основна увага приділялася методам зберігання даних про події об'єктів на сервері та їх синхронізації з локальним збереженням у браузері.

На першому етапі створено копію веб-сторінки із попередньої лабораторної роботи. Функціональність об’єкта було реалізовано з використанням чистого HTML, CSS та Javascript, без застосування сторонніх фреймворків. Це дозволило поглибити знання роботи з нативними технологіями.

Другий етап передбачав реалізацію двох способів зберігання подій на сервері:

1. **Негайне відправляння подій на сервер:**

- Події фіксувалися в режимі реального часу.
- Для кожної події зберігався порядковий номер та серверний час.
- Було проведено налаштування часу на сервері, щоб усунути розбіжності з локальним часом ПК.

2. **Акумуляція подій у LocalStorage:**

- Події зберігалися в LocalStorage з одночасним збереженням на сервері за першим способом.
- Після завершення роботи акумульовані дані було передано на сервер разом із фіксацією часу збереження.

Виявлено, що негайне збереження даних під час інтенсивного виконання подій може створювати додаткове навантаження на сервер, тоді як акумуляція в LocalStorage оптимізує роботу за рахунок зменшення кількості запитів.

Отримані результати показали, що ефективність обох способів збереження залежить від конкретного сценарію використання. Акумуляція даних у LocalStorage є оптимальним рішенням для роботи з великим обсягом подій, які не потребують негайного оброблення.

Результати лабораторної роботи були успішно розміщені на сервері, що демонструє їх функціональність у реальних умовах