SPRAWOZDANIE Z LABORATORIUM

Technologie Webowe (HTML/CSS/JavaScript)

## INFORMACJE PODSTAWOWE

**Imię i nazwisko studenta:Przemyslaw Chorążak**

**Numer albumu:28039**

**Grupa:3**

**Data wykonania laboratorium:**

**Numer laboratorium:**

**Temat laboratorium:React**

## **1. CEL LABORATORIUM**

Celem laboratorium było utrwalenie praktycznych umiejętności tworzenia aplikacji w React z wykorzystaniem komponentów funkcyjnych. W ramach ćwiczeń zrealizowałem aplikację „Menedżer Zadań”, w której zastosowałem: podział na komponenty, przekazywanie danych przez props (w tym children), zarządzanie stanem (useState), renderowanie list (map + key), formularze kontrolowane z walidacją, edycję elementów, filtrowanie i sortowanie danych oraz utrwalanie danych w localStorage. Dodatkowo wykorzystałem useEffect z obsługą ładowania/zapisu oraz przerwania żądań przez AbortController.

## **2. WYKONANE ZADANIA**

**Zadanie 1: Struktura projektu i komponenty**

**Status:** Wykonane w pełni  
**Opis realizacji:** Utworzyłem strukturę components/ oraz wydzieliłem logikę aplikacji na komponenty: Header, TaskForm, TaskList, TaskItem, FilterButtons, TaskStats oraz Card jako uniwersalny wrapper UI.

**Zadanie 2: Props / children / map + key**

**Status:** Wykonane w pełni  
**Opis realizacji:**

* Card przyjmuje children i opcjonalny title.
* Lista zadań renderowana przez tasks.map(...) z poprawnym key={task.id}.
* TaskItem przyjmuje propsy (m.in. title, completed, priority, category) z domyślnymi wartościami.

**Zadanie 3: Stan aplikacji, operacje na zadaniach, filtry i statystyki**

**Status:** Wykonane w pełni  
**Opis realizacji:** Użyłem useState do przechowywania listy zadań i ustawień widoku (filtr statusu, kategoria, wyszukiwanie, sortowanie). Dodałem akcje: dodanie zadania, przełączanie ukończenia, usuwanie, zmiana priorytetu oraz statystyki (liczba zadań, ukończone, pozostałe, procent).

**Zadanie 4: Formularz kontrolowany + walidacja + edycja**

**Status:** Wykonane w pełni  
**Opis realizacji:** Formularz dodawania jest kontrolowany (value + onChange) i waliduje tytuł (wymagany, min 3 znaki, max 100). TaskItem umożliwia edycję tytułu; Enter zapisuje, Escape anuluje.  
**Napotkane problemy:** Początkowo mało czytelny kontrast tekstu (biały na jasnym tle).  
**Rozwiązanie:** Ustawiłem globalny kolor tekstu w index.css (ciemny kolor) i dzięki temu UI jest czytelne.

**Zadanie 5: useEffect + localStorage / mock API + obsługa ładowania**

**Status:** Wykonane w pełni  
**Opis realizacji:** Dodałem warstwę tasksApi.js, która symuluje API (opóźnienia) i zapisuje/odczytuje dane z localStorage. Użyłem useEffect do wczytania danych po starcie i do automatycznego zapisu przy zmianie listy. Zaimplementowałem AbortController, aby przerwać operacje podczas unmount.  
**Napotkane problemy:** Błąd uruchomienia npm run dev w złym folderze (brak package.json).  
**Rozwiązanie:** Uruchamianie komend w folderze projektu (task-manager).

**Zadanie 6: Kategorie, wyszukiwanie, sortowanie**

**Status:** Wykonane w pełni  
**Opis realizacji:** Zadania posiadają kategorię i priorytet. Dodałem filtr kategorii (select), wyszukiwanie po tytule oraz sortowanie (domyślne / po priorytecie / alfabetycznie). W przypadku braku wyników wyświetlany jest komunikat.

## **3. FRAGMENTY KODU**

TasksApi.js

const delay = (ms, signal) =>

  new Promise((resolve, reject) => {

    const id = setTimeout(resolve, ms);

    if (signal) {

      signal.addEventListener("abort", () => {

        clearTimeout(id);

        reject(new DOMException("Aborted", "AbortError"));

      });

    }

  });

const LS\_KEY = "task\_manager\_tasks\_v1";

export async function fetchTasks({ signal } = {}) {

  await delay(600, signal);

  const raw = localStorage.getItem(LS\_KEY);

  if (!raw) return [];

  return JSON.parse(raw);

}

export async function saveTasks(tasks, { signal } = {}) {

  await delay(500, signal);

  localStorage.setItem(LS\_KEY, JSON.stringify(tasks));

  return true;

}

export function clearAllTasks() {

  localStorage.removeItem(LS\_KEY);

}

Header.jsx

import QuoteOfTheDay from "./QuoteOfTheDay";

export default function Header() {

  const today = new Date().toLocaleDateString("pl-PL", {

    weekday: "long",

    year: "numeric",

    month: "long",

    day: "numeric",

  });

  return (

    <header style={{ display: "grid", gap: 12 }}>

      <div style={{ display: "flex", alignItems: "baseline", justifyContent: "space-between", gap: 12, flexWrap: "wrap" }}>

        <h1 style={{ margin: 0 }}>📋 Menedżer Zadań</h1>

        <div style={{ opacity: 0.8 }}>{today}</div>

      </div>

      <QuoteOfTheDay />

    </header>

  );

}

// Przykład JavaScript

## **4. ZRZUTY EKRANU**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Zawartość wygenerowana przez AI może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, Czcionka, logo

Zawartość wygenerowana przez AI może być niepoprawna.

## **5. WNIOSKI I REFLEKSJE**

**Czego się nauczyłem:** praktycznego użycia useState i useEffect, zarządzania stanem formularza kontrolowanego oraz renderowania list w React. Nauczyłem się też organizowania aplikacji na komponenty oraz utrwalania danych w localStorage.

**Co było najtrudniejsze:** spięcie logiki filtrowania/sortowania w czytelny sposób oraz poprawna obsługa ładowania/zapisu danych z cleanup (AbortController).

**Co chciałbym przećwiczyć:** testowanie komponentów, lepsze zarządzanie stanem (np. Context / reducer) oraz dopracowanie UI w oparciu o bibliotekę komponentów.

## **6. SAMOOCENA**

Ocena własnego zaangażowania: ☐ Bardzo wysokie

Procent wykonanych zadań: 100%

Dodatkowe uwagi: aplikacja działa lokalnie pod Vite, zapisuje zadania w localStorage, posiada filtrowanie, wyszukiwanie, sortowanie, edycję i pobieranie cytatu dnia

Data wypełnienia sprawozdania:8.02.2026

Podpis studenta:Przemysław Chorążak