2PRO - Projekt (logowanie) wymagania

Wymagania techniczne (obowiązkowe) projektu na ocenę z dodatkowymi wyjaśnieniami, przykładami i dokładnym kryterium oceniania:

Wymagania techniczne (obowiązkowe):

1. **Temat projektu**
   - Każdy uczeń samodzielnie wybiera temat projektu – może to być prosta aplikacja użytkowa (np. lista zadań, katalog filmów, rejestr zajęć) lub coś kreatywnego (np. mini-gra przeglądarkowa).
   - W projekcie musi się znaleźć **nazwa**, **krótkie wprowadzenie** oraz **cel aplikacji**.
   - Wskazane jest, aby temat był realny i funkcjonalnie dopasowany do możliwości ucznia.
2. **Obsługa wielu użytkowników**
   - Aplikacja musi umożliwiać rejestrację oraz logowanie wielu użytkowników.
   - Każdy użytkownik powinien mieć przypisane konto (login i hasło).
   - Dane użytkowników trzeba przechowywać w lokalnej bazie danych.
   - Rejestracja musi zawierać:
     - pole do wpisania loginu,
     - pole do wpisania hasła,
     - pole do powtórzenia hasła (sprawdzenie zgodności),
     - sprawdzenie, czy użytkownik o podanym loginie już istnieje,
     - hasło musi być zaszyfrowane za pomocą funkcji `md5()` (nie zapisujemy hasła w postaci jawnej).
   - Logowanie:

    - porównuje dane z pliku/bazy i umożliwia zalogowanie tylko poprawnym użytkownikom.

3. **System ról użytkowników**
    - Po zalogowaniu aplikacja rozpoznaje, czy użytkownik ma rolę `user` czy `admin`.
    - Rola `admin` ma dostęp do funkcji niedostępnych dla zwykłych użytkowników, np.:
        - podgląd wszystkich kont,
        - edycja/usuwanie kont,
        - dodatkowe dane w dashboardzie.
    - Przykład rekordu użytkownika w pliku `.json`:

```
{
  "login": "admin",
  "password": "b59c67bf196a4758191e42f76670ceba",
  "role": "admin"
}
```

4. **Logowanie i sesja**
    - Po poprawnym logowaniu tworzona jest sesja lub zapis do ciasteczka, które pozwala rozpoznać użytkownika.
    - Zalogowany użytkownik powinien być przekierowany do odpowiedniego **dashboardu**.
    - W przypadku wyłączenia przeglądarki lub wylogowania, sesja powinna wygasać.
5. **Zabezpieczenie dostępu**
    - Jeśli użytkownik niezalogowany próbuje wejść na adres `/dashboard`, powinien zostać automatycznie przekierowany na stronę główną (`main`) lub stronę logowania.
    - Funkcje dostępne tylko dla admina (np. `/admin-panel`) powinny być zabezpieczone dodatkową kontrolą roli.
6. **Dashboard (panel użytkownika)**

- Jest to główna strona działania aplikacji, widoczna tylko po zalogowaniu.
- Powinien zawierać podstawową funkcjonalność zgodną z tematem projektu, np.:
  - lista zadań (dodawanie, edycja, usuwanie),
  - formularz kontaktowy,
  - panel ogłoszeń.
- Zwykły użytkownik widzi tylko własne dane.
- Administrator ma dodatkowe możliwości (np. przegląd danych innych użytkowników, usuwanie kont itp.).

7. **Wylogowanie**
   - Aplikacja musi zawierać przycisk do wylogowania.
   - Po kliknięciu, sesja lub ciasteczko powinno zostać usunięte.
   - Użytkownik powinien zostać przeniesiony na stronę logowania lub główną.

Ocena projektu – skala punktowa

Kryterium	Ocena **dopuszczająca (2)**	Ocena **dostateczna (3)**	Ocena **dobra (4)**	Ocena **bardzo dobra (5)**
Rejestracja i logowanie	Działa tylko logowanie	Działa logowanie i prosta rejestracja	Zgodne z zajęciami, hasło jako `md5`	Dodatkowe walidacje, unikalność loginu
Obsługa wielu użytkowników	1 użytkownik	Wielu, bez osobnych danych	Wielu z osobnymi danymi	Dobrze odseparowane konta *
System ról (`user`, `admin`)	Brak	Rola admin bez funkcji	Admin ma inny widok	Admin może zarządzać kontami
Dashboard	Statyczna zawartość	Statyczna zawartość	Działa część funkcji	Zgodny z tematem i roli

Zabezpieczenie dostępu i sesje	Zabezpieczenie dashboardu	Zabezpieczenie dashboardu	Zabezpieczenie dashboardu	Zabezpieczenie dla roli admina i Zabezpieczenie dashboardu
Wylogowanie	Brak	Działa	Poprawne	Czyści sesję i przekierowuje

(*)Przez **„dobrze odseparowane konta”** mam na myśli to, że:

- każdy użytkownik **widzi i może modyfikować tylko swoje dane**,
- nie ma możliwości „podglądania” cudzych informacji (np. zadań, ogłoszeń, wypożyczeń itp.),
- aplikacja **rozpoznaje zalogowanego użytkownika** i działa na danych przypisanych do jego konta (np. `userId`, `login`, itp.),
- dane są **powiązane z konkretnym użytkownikiem**, np. w pliku lub bazie każdy wpis (np. zadanie) ma pole `ownerId` lub `createdBy`.

Przykład:

Jeśli dwóch użytkowników zaloguje się:

- użytkownik `janek` widzi tylko swoje zadania,
- użytkownik `ania` widzi tylko swoje ogłoszenia,
- a `admin` może przeglądać wszystko (jeśli mu na to pozwolimy).

Należy sporządzić dokument z nazwą projektu i jego funkcjonalnościami z podziałem na użytkowników