

Wymagania edukacyjne

Zaawansowane aplikacje webowe/Projektowanie zaawansowanych aplikacji webowych Technik programista

Sposoby sprawdzania i oceniania osiągnięć edukacyjnych uczniów. Ocenie podlega zarówno wiedza teoretyczna, jak i nabyte w trakcie nauki umiejętności.

Oceniane są:

1. Ćwiczenia, zadania wykonywane na lekcji

Ocenie podlega: wykonanie wszystkich poleceń zgodnie z treścią, stopień samodzielności wykonywania zadania, pilność, końcowy efekt pracy (jakość pracy), umiejętność pracy w zespole, terminowość – warunek otrzymania oceny dobrej i wyższej.

2. Aktywność podczas pracy na lekcji.

Ocenie podlega: aktywność ucznia w czasie zajęć, stopień zaangażowania podczas wykonywania zajęć, zainteresowanie tematem lekcji.

3. Kartkówki, testy, sprawdziany pisemne lub praktyczne.

Ocena z prac pisemnych zgodna ze Statutem Szkoły

4. Zadania dodatkowe, prace projektowe

5. Skala ocen:

Celujący (6): 100%

Bardzo dobry (5): 90% – 99%

Dobry (4): 75% – 89%

Dostateczny (3): 51% – 74%

Dopuszczający (2): 30% – 50%

Niedostateczny (1): 0% – 29%

Wymagania na poszczególne oceny				
Dopuszczający	Dostateczny	Dobry	Bardzo dobry	Celujący
Środowisko programistyczne				
- Zna podstawowe środowiska programistyczne dla aplikacji webowych - Wie jakie języki programowania wykorzystywane są do tworzenia aplikacji webowych - Potrafi otworzyć folder z projektem w edytorze i odnaleźć wbudowany terminal (wiersz poleceń). - Potrafi zainstalować zalecaną wtyczkę do edytora wspomagającą formatowanie kodu (np. Prettier).	- Wie jak skonfigurować środowisko pracy do tworzenia aplikacji webowych - Wie jak zainstalować biblioteki i wykorzystać menadżer pakietów - Wie jak zainstalować środowisko uruchomieniowe Node.js na systemie operacyjnym. - Potrafi pobrać i zainstalować wszystkie wymagane zależności dla pobranego projektu za pomocą polecenia npm install. - Umie zainstalować nową, pojedynczą bibliotekę (np. Bootstrap) przy użyciu wiersza poleceń.	- Zna najczęściej używane biblioteki - Potrafi samodzielnie wygenerować nowy, czysty projekt za pomocą oficjalnych narzędzi CLI (np. npm create vite@latest lub ng new).	- Wie jak skonfigurować kompletne środowisko pracy ze wszystkimi niezbędnymi bibliotekami i frameworkami - Wie jak wykorzystać dokumentację techniczną do samodzielnej konfiguracji środowiska pod nietypową bibliotekę.	- Wie jak wykorzystać dokumentację by wykorzystać dodatkowe możliwości środowiska programistycznego i bibliotek - Samodzielnie konfiguruje zaawansowane narzędzia automatyzujące pracę (np. niestandardowe skrypty startowe lub systemy kontroli jakości kodu - linter).
REACT				
- Wie jak uruchomić gotowy projekt i lokalny serwer deweloperski (npm run dev lub npm start).	- Potrafi zapisać prosty komponent funkcyjny przy użyciu funkcji strzałkowej (const MyComponent = () => {}).	- Potrafi przypisać funkcję strzałkową do obsługi zdarzeń interfejsu (np. onClick={() => handleStore()} lub jako osobną zmienną).	- Potrafi połączyć aplikację z zewnętrznym API i asynchronicznie pobrać dane wewnątrz hooka useEffect.	- Buduje własne, reużywalne hooki (custom hooks) zapisane w postaci funkcji strzałkowych w celu

- Zna podstawową strukturę plików projektu i potrafi wskazać główny komponent App. - Potrafi zmodyfikować statyczny tekst lub znaczniki HTML wewnątrz składni JSX.	- Wie jak przekazać proste dane do komponentu za pomocą właściwości (props). - Wykorzystuje funkcję strzałkową wewnątrz metody .map() do dynamicznego wygenerowania listy elementów (np. menu lub tabeli). - Potrafi użyć hooka useState do prostego zarządzania stanem komponentu (np. zmiana tekstu, licznik).	☑ Stosuje operatory logiczne && oraz operatory ternarne (? :) do warunkowego renderowania elementów na stronie. - Potrafi stworzyć formularz kontrolowany, aktualizując stan aplikacji przy każdej zmianie w polu tekstowym (onChange). - Wie jak podzielić aplikację na podstrony i skonfigurować podstawowe ścieżki (Routing).	- Wykorzystuje funkcje strzałkowe do przekazywania callbacków w górę drzewa komponentów (podnoszenie stanu / lifting state up). - Zna kody statusów HTTP i potrafi obsłużyć błędy połączenia z serwerem, wyświetlając odpowiedni komunikat użytkownikowi. - Wie jak zorganizować strukturę plików projektu zgodnie z dobrymi praktykami (podział na komponenty, widoki i usługi).	wydzielenia logiki biznesowej. - Samodzielnie diagnozuje błędy renderowania i ostrzeżenia (warnings) w konsoli deweloperskiej przeglądarki.
ANGULAR				
- Wie jak uruchomić gotowy projekt i serwer deweloperski (ng serve). - Potrafi zmienić tekst lub prosty znacznik HTML w pliku .component.html. - Rozróżnia, który plik odpowiada za wygląd (HTML), a który za logikę (TS).	- Potrafi wyświetlić zmienną z kodu w szablonie HTML za pomocą nawiasów {{ zmienna }} (interpolacja). - Wie jak podpiąć zdarzenie kliknięcia pod przycisk przy użyciu składni (click)="funkcja()". - Potrafi wyświetlić listę danych (np. tabelę produktów) za pomocą pętli @for. - Wie jak użyć instrukcji @if do prostego ukrywania lub	- Potrafi zastosować konstrukcję @if / @else do przełączania widoków (np. wyświetlenie tekstu „Brak produktów”, gdy lista jest pusta). - Wie jak wykorzystać instrukcję @switch wraz z @case do wyświetlania różnych elementów interfejsu w zależności od stanu (np. status zamówienia: nowe, w realizacji, wysłane).	- Potrafi połączyć aplikację z internetowym źródłem danych (API) i pobrać z niego informacje. - Wie jak oddzielić kod odpowiedzialny za pobieranie danych od kodu wyświetlającego stronę, tworząc tzw. Serwis. - Wykorzystuje instrukcje @if oraz @switch, aby pokazać komunikat o ładowaniu danych lub błędzie połączenia.	- Potrafi samodzielnie dodać walidację do formularza i dynamicznie wyświetlać komunikaty o błędach za pomocą @if. - Samodzielnie diagnozuje i rozwiązuje błędy w konsoli przeglądarki na podstawie oficjalnej dokumentacji frameworka.

	pokazywania elementu na stronie (np. komunikatu o błędzie).	- Potrafi stworzyć prosty formularz ([[ngModel]]) i przechwycić dane wpisane przez użytkownika.	- Potrafi ukryć lub pokazać elementy menu (np. profil użytkownika) w zależności od tego, czy użytkownik jest zalogowany.	
BOOTSTRAP				
- Wie, jak połączyć Bootstrap z projektem webowym za pomocą linków z Internetu (CDN). - Potrafi odnaleźć w dokumentacji, skopiować i wyświetlić pojedynczy, prosty element, taki jak przycisk (btn) lub ikona. - Rozróżnia podstawowe klasy odpowiedzialne za wygląd (np. wie, że klasa btn-primary zmienia kolor przycisku na niebieski).	- Wie jak działa system siatki - Potrafi ułożyć elementy w kolumnach i rzędach przy użyciu podstawowego systemu siatki (row, col-*). - Wykorzystuje gotowe klasy do stworzenia czytelnej tabeli z danymi (table, table-striped, table-bordered) – np. do wyświetlenia cennika lub listy użytkowników. - Potrafi estetycznie sformatować pola formularza (np. ramki pól tekstowych, przyciski akcji) za pomocą klasy form-control.	- Potrafi zaprojektować układ strony, który dostosowuje się do ekranu smartfona i komputera za pomocą klas responsywnych (np. col-12 col-md-6). - Wie jak kontrolować odstępy między elementami aplikacji, wykorzystując klasy marginesów i dopełnień (Spacing utilities: np. mt-3, pb-2, mx-auto). - Wdraża górny pasek nawigacyjny (Navbar) z logo i linkami oraz ramki z powiadomieniami (Alerts – np. zielona ramka sukcesu po zapisaniu danych). - Potrafi zintegrować komponenty Bootstrapa z logiką aplikacji	- Potrafi precyzyjnie pozycjonować elementy (np. wyrównywać tekst pionowo do środka, rozsuwać przyciski na przeciwległe boki) za pomocą narzędzia Flexbox (d-flex, justify-content-between, align-items-center). - Implementuje działające elementy interaktywne wymagające skryptów JavaScript: karuzelę (pokaz slajdów / carousel) z automatycznie przewijanymi zdjęciami oraz wyskakujące okienka (Modal np. jako potwierdzenie przed usunięciem konta). - Zmienia domyślną kolorystykę komponentów, nagłówek i tabel, korzystając z pełnej palety klas kontekstowych (np. bg-light, text-muted, border-danger).	- Potrafi nadpisać domyślne style Bootstrapa za pomocą własnego pliku CSS lub zmiennych, dbając o unikalny wygląd aplikacji. - Samodzielnie buduje zaawansowane, niestandardowe układy interfejsu (np. boczny panel nawigacyjny – Sidebar, pulpit nawigacyjny – Dashboard) w oparciu o dokumentację techniczną.

Exspress.js				
- Wie jak uruchomić gotowy program serwera za pomocą terminala. - Potrafi znaleźć w kodzie numer portu, na którym działa serwer. - Potrafi zmienić tekst, który serwer wyświetla po wejściu na stronę główną.	- Potrafi stworzyć proste podstrony na serwerze (np. /o-nas, /kontakt) za pomocą metody GET. - Wie jak przesłać z serwera do przeglądarki dane w formacie JSON (np. prostą listę produktów). - Potrafi odebrać proste dane tekstowe, które użytkownik wpisał i wysłał z formularza na stronie.	- Potrafi stworzyć stronę, która reaguje na zmienny parametr w adresie URL (np. wyświetla dane konkretnego użytkownika po jego ID). - Wie jak połączyć serwer z bazą danych, aby pobrać z niej lub zapisać w niej informacje. - Stosuje podstawowe numery statusów odpowiedzi (np. wie, kiedy serwer powinien zwrócić błąd 404 Brak strony, a kiedy 200 OK).	- Potrafi napisać kompletny program obsługujący bazę danych (dodawanie, wyświetlanie, edycja i usuwanie wpisów). - Wie jak zabezpieczyć wybrane podstrony serwera tak, aby dostęp do nich miały tylko zalogowane osoby. - Potrafi połączyć stworzony serwer z aplikacją frontendową (w Angularze lub Reactcie), rozwiązując problem blokowania połączeń (CORS).	- Potrafi zabezpieczyć hasła użytkowników w bazie danych przed kradzieżą (poprzez ich zaszyfrowanie). - Samodzielnie naprawia błędy w działaniu serwera, korzystając z komunikatów w terminalu i dokumentacji technicznej.