Dokumentacja do pliku add_page.php

Cel pliku:

Plik add_page.php umożliwia administratorowi dodawanie nowych podstron do bazy danych. Funkcjonalność ta dostępna jest wyłącznie dla użytkowników zalogowanych z uprawnieniami administratora.

Działanie pliku:

1. Sprawdzenie sesji użytkownika

- Plik rozpoczyna sesję za pomocą session_start().

- Sprawdza, czy użytkownik jest zalogowany, poprzez obecność zmiennej $_SESSION['user_id'].

2. Weryfikacja uprawnień

- Jeżeli użytkownik jest zalogowany, wykonywane jest zapytanie do bazy danych w celu pobrania jego roli (rodzaj).

- Jeżeli rola to admin, użytkownik uzyskuje dostęp do strony. W przeciwnym wypadku jest przekierowywany na stronę logowania (login.php).

3. Dodawanie nowej podstrony

- Jeżeli użytkownik jest administratorem, może wypełnić formularz i dodać nową podstronę do bazy danych.

- Formularz zawiera dwa pola: tytuł i treść podstrony.

- Po przesłaniu formularza dane są zapisywane w tabeli "podstrony" w bazie danych.

- Po zapisaniu danych użytkownik jest przekierowywany do strony z listą podstron (podstrony.php).

Nagłówek HTML:

- Formularz zawiera pola do wpisania tytułu i treści podstrony.

- Formularz jest obsługiwany metodą POST, a dane są przesyłane do tego samego pliku (add_page.php).

Uwagi:

- Tylko użytkownicy z rolą 'admin' mają dostęp do tej strony.

- W przypadku nieautoryzowanego dostępu następuje przekierowanie do strony logowania.

Dokumentacja do pliku admin.php

Cel pliku:

Plik admin.php służy do zarządzania produktami w panelu administracyjnym sklepu motocyklowego. Administrator lub pracownik może dodawać nowe produkty, edytować istniejące, usuwać je oraz przeglądać listę produktów. Formularz umożliwia przesyłanie zdjęć produktów oraz przypisanie ich do odpowiednich kategorii.

Działanie pliku:

1. Sprawdzenie sesji użytkownika:

   - Plik rozpoczyna sesję za pomocą session_start().

   - Sprawdza, czy użytkownik jest zalogowany poprzez obecność zmiennej $_SESSION['user_id'].

   - Jeżeli użytkownik jest zalogowany, sprawdzana jest jego rola. Jeśli rola to 'admin' lub 'pracownik', użytkownik uzyskuje dostęp do panelu.

2. Pobranie produktów z bazy danych:

   - Plik wykonuje zapytanie SQL, które pobiera wszystkie produkty z bazy danych wraz z kategoriami produktów.

3. Dodawanie nowego produktu:

   - Jeżeli użytkownik wypełni formularz, dane są przesyłane metodą POST.

   - Formularz zawiera pola do wprowadzenia nazwy, kategorii, ceny, opisu oraz zdjęcia produktu.

   - Jeżeli formularz jest poprawnie wypełniony, dane produktu są zapisywane w tabeli "products", a użytkownik jest przekierowywany do strony `admin.php`.

4. Wyświetlanie listy produktów:

   - Produkty są wyświetlane w tabeli, zawierającej kolumny: ID, nazwę, typ kategorii, zdjęcie, cenę, opis oraz opcje (edytuj, usuń, zobacz).

   - Każdy produkt ma przypisane odpowiednie opcje umożliwiające edycję, usunięcie lub podgląd szczegółów produktu.

Nagłówek HTML:

- Strona zawiera nagłówek z tytułem panelu administracyjnego oraz linkami do innych sekcji zarządzania (użytkownikami, kategoriami, zamówieniami, podstronami, produktami, dostawami).

- Przycisk "Dodaj nowy produkt" umożliwia przejście do formularza dodawania nowego produktu.

Formularz dodawania produktu:

- Formularz zawiera pola do wprowadzenia nazwy produktu, kategorii, ceny, opisu oraz zdjęcia produktu. Po jego wypełnieniu, dane są przesyłane metodą POST do tego samego pliku.

Lista produktów:

- Produkty są wyświetlane w tabeli z możliwością edycji, usunięcia lub podglądu.

Uwagi:

- Tylko użytkownicy z rolą 'admin' lub 'pracownik' mają dostęp do tej strony.

- W przypadku niekompletnych danych formularza użytkownik otrzymuje komunikat o błędzie.

Dokumentacja do pliku db.php

Cel pliku:

Plik db.php służy do nawiązania połączenia z bazą danych MySQL. Zawiera konfigurację połączenia zarówno za pomocą obiektu mysqli, jak i PDO, umożliwiając korzystanie z obu metod w innych częściach aplikacji.

Działanie pliku:

1. **Połączenie z bazą danych za pomocą mysqli:**

   - Zmienna `$host` zawiera adres serwera bazy danych (w tym przypadku 'localhost').

   - Zmienna `$db` zawiera nazwę bazy danych (w tym przypadku 'm10280_motocykle_skep').

   - Zmienna `$user` zawiera nazwę użytkownika bazy danych (w tym przypadku 'root').

   - Zmienna `$pass` zawiera hasło użytkownika bazy danych (w tym przypadku puste, co oznacza brak hasła w przypadku lokalnego połączenia).

   - Obiekt `$conn` służy do nawiązania połączenia z bazą danych przy użyciu klasy `mysqli`.

2. **Połączenie z bazą danych za pomocą PDO:**

   - Obiekt `$pdo` jest tworzony przy użyciu klasy `PDO` z odpowiednimi parametrami: `mysql:host`, `dbname`, użytkownik, hasło.

   - Ustawienie atrybutu `PDO::ATTR_ERRMODE` na `PDO::ERRMODE_EXCEPTION` pozwala na rzucanie wyjątków w przypadku błędów połączenia.

   - W przypadku niepowodzenia w połączeniu, zostaje rzucony wyjątek, a użytkownik otrzymuje komunikat o błędzie z opisem problemu.

Uwagi:

- Połączenie zarówno za pomocą mysqli, jak i PDO jest zrealizowane, ale w zależności od dalszej implementacji, jeden z tych sposobów może być używany w aplikacji (zalecane jest korzystanie z PDO dla lepszej obsługi wyjątków i przenośności kodu).

- W przypadku błędu połączenia z bazą danych, skrypt wyświetli komunikat o błędzie i zakończy działanie.

Dokumentacja do pliku delete_category.php

Cel pliku:

Plik `delete_category.php` służy do usuwania kategorii z bazy danych. Umożliwia to administratorom oraz pracownikom zarządzającym kategoriami produktów w sklepie motocyklowym usunięcie niepotrzebnych kategorii. Po usunięciu, użytkownik jest przekierowywany do strony zarządzania kategoriami.

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

   - Rozpoczynana jest sesja za pomocą `session_start()`.

   - Sprawdzane jest, czy użytkownik jest zalogowany poprzez obecność zmiennej `$_SESSION['user_id']`.

   - Jeżeli użytkownik jest zalogowany, jego dane są pobierane z bazy danych (nazwa użytkownika, email, rodzaj) przy pomocy zapytania SQL.

   - Następnie sprawdzana jest rola użytkownika. Tylko użytkownicy z rolą `admin` lub `pracownik` mają dostęp do tej strony.

2. **Połączenie z bazą danych:**

   - Używane jest połączenie PDO z bazą danych (mysql:host), w tym przypadku do bazy o kodowaniu `utf8`.

- Jeśli połączenie z bazą danych nie powiedzie się, użytkownik otrzyma komunikat o błędzie w formacie JSON i skrypt zakończy działanie.

3. **Usuwanie kategorii:**

- Sprawdzane jest, czy parametr `id` został przekazany za pomocą metody GET.

- Jeśli `id` jest dostępne, jego wartość jest konwertowana na liczbę całkowitą przy pomocy funkcji `intval()`.

- Następnie wykonane jest zapytanie SQL usuwające kategorię z tabeli `categories`, której identyfikator odpowiada przekazanemu `id`.

4. **Przekierowanie:**

- Po wykonaniu zapytania usuwającego kategorię, użytkownik jest przekierowywany na stronę zarządzania kategoriami (`kategorie.php`).

Uwagi:

- Skrypt zakłada, że użytkownik jest zalogowany i posiada odpowiednie uprawnienia (rola `admin` lub `pracownik`), aby móc usuwać kategorie.

- W przypadku błędu połączenia z bazą danych, użytkownik otrzymuje komunikat o błędzie w formacie JSON.

- Skrypt wymaga przekazania parametru `id` w adresie URL (np. `delete_category.php?id=1`) w celu usunięcia odpowiedniej kategorii.

Dokumentacja do pliku delete_order.php

Cel pliku:

Plik `delete_order.php` służy do usuwania zamówienia z bazy danych. Umożliwia administratorom oraz pracownikom sklepu usunięcie zamówienia, w tym powiązanych z nim pozycji zamówienia. Po pomyślnym usunięciu, użytkownik zostaje przekierowany do panelu administracyjnego.

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

- Skrypt rozpoczyna sesję za pomocą `session_start()`.

- Następnie sprawdza, czy użytkownik jest zalogowany, sprawdzając obecność zmiennej `$_SESSION['user_id']`.

- Jeśli użytkownik jest zalogowany, jego dane (nazwa użytkownika, email, rodzaj) są pobierane z bazy danych za pomocą zapytania SQL.

- Weryfikowana jest rola użytkownika. Użytkownicy z rolą `admin` lub `pracownik` mają dostęp do tego skryptu.

2. **Sprawdzenie uprawnień użytkownika:**

- Jeżeli użytkownik nie posiada wymaganych uprawnień (`admin` lub `pracownik`), jest przekierowywany do strony logowania (`login.php`).

3. **Sprawdzanie obecności identyfikatora zamówienia:**

- Skrypt sprawdza, czy parametr `id` (identyfikator zamówienia) jest przekazany w adresie URL.

- Jeśli parametr `id` jest niedostępny, skrypt wyświetla komunikat "Brak ID zamówienia" i kończy działanie.

4. **Usuwanie pozycji zamówienia:**

- Jeśli `id` zamówienia jest dostępne, skrypt wykonuje zapytanie SQL, które usuwa wszystkie pozycje powiązane z danym zamówieniem w tabeli `order_items`.

5. **Usuwanie zamówienia:**

- Następnie skrypt wykonuje zapytanie SQL usuwające zamówienie z tabeli `orders`, którego identyfikator odpowiada przekazanemu `id`.

6. **Przekierowanie:**

- Po usunięciu zamówienia i jego pozycji, użytkownik zostaje przekierowany na stronę panelu administracyjnego (`admin.php`).

Uwagi:

- Skrypt wymaga przekazania parametru `id` w adresie URL (np. `delete_order.php?id=1`) w celu usunięcia odpowiedniego zamówienia.

- Tylko użytkownicy z rolą `admin` lub `pracownik` mają dostęp do tego skryptu.

- Po pomyślnym usunięciu zamówienia, użytkownik jest przekierowywany do panelu administracyjnego.

- Skrypt nie obsługuje sytuacji, gdy parametr `id` jest nieprawidłowy lub nieistnieje w bazie danych.

Dokumentacja do pliku delete_page.php

Cel pliku:

Plik `delete_page.php` umożliwia usunięcie podstrony z bazy danych na podstawie jej identyfikatora (`id`). Skrypt zapewnia, że tylko administratorzy mają dostęp do tej funkcji. Po usunięciu podstrony, użytkownik jest przekierowywany na stronę zarządzania podstronami.

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

   - Skrypt rozpoczyna sesję za pomocą `session_start()`.

   - Sprawdzana jest obecność zmiennej `$_SESSION['user_id']`, aby zweryfikować, czy użytkownik jest zalogowany.

   - Jeśli użytkownik jest zalogowany, jego dane (rodzaj użytkownika) są pobierane z bazy danych za pomocą zapytania SQL.

   - Jeśli użytkownik ma rolę `admin`, przypisywana jest zmienna `$isAdmin` jako `true`.

2. **Weryfikacja uprawnień użytkownika:**

   - Jeśli użytkownik nie ma uprawnień administratora, zostaje przekierowany na stronę logowania (`login.php`).

3. **Sprawdzanie obecności identyfikatora podstrony:**

   - Skrypt sprawdza, czy parametr `id` (identyfikator podstrony) został przekazany w adresie URL.

   - Jeśli parametr `id` jest obecny, wykonuje zapytanie SQL, aby pobrać dane podstrony z tabeli `podstrony`.

4. **Usuwanie podstrony:**

   - Jeśli podstrona o danym `id` istnieje w bazie danych, skrypt wykonuje zapytanie SQL, aby usunąć tę podstronę z tabeli `podstrony`.

   - Po pomyślnym usunięciu podstrony, użytkownik jest przekierowywany na stronę zarządzania podstronami (`podstrony.php`).

5. **Obsługa błędów:**

   - Jeśli podstrona o danym `id` nie istnieje, skrypt wyświetla komunikat: "Podstrona o takim ID nie istnieje".

- Jeśli parametr `id` nie jest przekazany w adresie URL, skrypt wyświetla komunikat: "Brak ID podstrony do usunięcia".

Uwagi:

- Skrypt wymaga przekazania parametru `id` w adresie URL (np. `delete_page.php?id=1`) w celu usunięcia odpowiedniej podstrony.

- Tylko użytkownicy z rolą `admin` mają dostęp do tego skryptu.

- Po pomyślnym usunięciu podstrony, użytkownik zostaje przekierowany na stronę zarządzania podstronami.

- Skrypt nie obsługuje sytuacji, gdy parametr `id` jest nieprawidłowy lub nieistnieje w bazie danych.

Dokumentacja do pliku delete_product.php

Cel pliku:

Plik `delete_product.php` umożliwia usunięcie produktu z bazy danych na podstawie jego identyfikatora (`id`). Skrypt zapewnia, że tylko użytkownicy z rolą `admin` lub `pracownik` mogą usunąć produkt. Po pomyślnym usunięciu produktu, użytkownik zostaje przekierowany na stronę panelu administracyjnego (`admin.php`).

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

   - Skrypt rozpoczyna sesję za pomocą `session_start()`.

   - Sprawdzana jest obecność zmiennej `$_SESSION['user_id']`, aby zweryfikować, czy użytkownik jest zalogowany.

   - Jeśli użytkownik jest zalogowany, jego dane (w tym rola użytkownika) są pobierane z bazy danych za pomocą zapytania SQL.

   - Jeśli użytkownik ma rolę `admin` lub `pracownik`, zmienna `$isAllowed` jest ustawiana na `true`, co daje mu dostęp do dalszego działania.

2. **Weryfikacja uprawnień użytkownika:**

   - Jeśli użytkownik nie ma odpowiednich uprawnień (nie jest administratorem ani pracownikiem), skrypt nie wykonuje dalszych działań.

3. **Sprawdzanie obecności identyfikatora produktu:**

- Skrypt sprawdza, czy parametr `id` (identyfikator produktu) został przekazany w adresie URL.

- Jeśli parametr `id` jest obecny, wykonuje zapytanie SQL, aby usunąć produkt o danym `id` z tabeli `products`.

4. **Usuwanie produktu:**

- Jeśli parametr `id` został przekazany, skrypt wykonuje zapytanie SQL, aby usunąć produkt o podanym `id` z bazy danych.

- Po pomyślnym usunięciu produktu, użytkownik zostaje przekierowany na stronę panelu administracyjnego (`admin.php`).

5. **Obsługa błędów:**

- Jeśli parametr `id` nie jest przekazany w adresie URL, skrypt wyświetla komunikat: "Brak ID produktu" i kończy działanie.

Uwagi:

- Skrypt wymaga przekazania parametru `id` w adresie URL (np. `delete_product.php?id=1`) w celu usunięcia odpowiedniego produktu.

- Tylko użytkownicy z rolą `admin` lub `pracownik` mają dostęp do tego skryptu.

- Po pomyślnym usunięciu produktu, użytkownik zostaje przekierowany na stronę panelu administracyjnego (`admin.php`).

- Skrypt nie obsługuje sytuacji, gdy parametr `id` jest nieprawidłowy lub nieistnieje w bazie danych.

Dokumentacja do pliku delete_shipping_method.php

Cel pliku:

Plik `delete_shipping_method.php` umożliwia usunięcie metody dostawy z bazy danych na podstawie jej identyfikatora (`id`). Skrypt zapewnia, że tylko użytkownicy z rolą `admin` mogą usunąć metodę dostawy. Po pomyślnym usunięciu metody dostawy, użytkownik zostaje przekierowany na stronę zarządzania dostawami (`dostawy.php`).

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

- Skrypt rozpoczyna sesję za pomocą `session_start()`.

- Sprawdzana jest obecność zmiennej `$_SESSION['user_id']`, aby zweryfikować, czy użytkownik jest zalogowany.

- Jeśli użytkownik jest zalogowany, jego dane (w tym rola użytkownika) są pobierane z bazy danych za pomocą zapytania SQL.

- Jeśli użytkownik ma rolę `admin`, zmienna `$isAdmin` jest ustawiana na `true`, co daje mu dostęp do dalszego działania.

2. **Weryfikacja uprawnień użytkownika:**

- Jeśli użytkownik nie ma roli `admin`, skrypt przekierowuje go do strony logowania (`login.php`) i kończy działanie.

3. **Sprawdzanie obecności identyfikatora metody dostawy:**

- Skrypt sprawdza, czy parametr `id` (identyfikator metody dostawy) został przekazany w adresie URL.

- Jeśli parametr `id` jest obecny, wykonuje zapytanie SQL, aby usunąć metodę dostawy o danym `id` z tabeli `Shipping_methods`.

4. **Usuwanie metody dostawy:**

- Jeśli parametr `id` został przekazany, skrypt wykonuje zapytanie SQL, aby usunąć metodę dostawy o podanym `id` z bazy danych.

- Po pomyślnym usunięciu metody dostawy, użytkownik zostaje przekierowany na stronę zarządzania dostawami (`dostawy.php`).

5. **Obsługa błędów:**

- Jeśli parametr `id` nie jest przekazany w adresie URL, skrypt przekierowuje użytkownika na stronę zarządzania dostawami (`dostawy.php`).

Uwagi:

- Skrypt wymaga przekazania parametru `id` w adresie URL (np. `delete_shipping_method.php?id=1`) w celu usunięcia odpowiedniej metody dostawy.

- Tylko użytkownicy z rolą `admin` mają dostęp do tego skryptu.

- Po pomyślnym usunięciu metody dostawy, użytkownik zostaje przekierowany na stronę zarządzania dostawami (`dostawy.php`).

- Skrypt nie obsługuje sytuacji, gdy parametr `id` jest nieprawidłowy lub nieistnieje w bazie danych.

Dokumentacja do pliku dostawy.php

Cel pliku:

Plik `dostawy.php` jest częścią panelu administracyjnego sklepu, który umożliwia zarządzanie metodami dostawy. Użytkownik może przeglądać istniejące metody dostawy, dodawać nowe metody oraz edytować lub usuwać istniejące. Dostęp do tej strony mają tylko użytkownicy z rolą `admin`.

Działanie pliku:

1. **Sprawdzenie sesji użytkownika:**

   - Skrypt rozpoczyna sesję za pomocą `session_start()`.

   - Sprawdzana jest obecność zmiennej `$_SESSION['user_id']`, aby zweryfikować, czy użytkownik jest zalogowany.

   - Jeśli użytkownik jest zalogowany, jego dane (w tym rola użytkownika) są pobierane z bazy danych za pomocą zapytania SQL.

   - Jeśli użytkownik ma rolę `admin`, zmienna `$isAdmin` jest ustawiana na `true`, co pozwala na dostęp do strony. W przeciwnym razie użytkownik jest przekierowywany do strony logowania (`login.php`).

2. **Dodawanie nowej metody dostawy:**

   - Jeśli metoda POST jest wywoływana (np. formularz został wysłany), skrypt pobiera dane z formularza (`name` i `cost`) i zapisuje je w tabeli `Shipping_methods` w bazie danych.

   - Po pomyślnym dodaniu metody dostawy, użytkownik jest przekierowywany na stronę `dostawy.php` (aby odświeżyć listę metod).

3. **Wyświetlanie metod dostawy:**

   - Skrypt wykonuje zapytanie SQL, aby pobrać wszystkie metody dostawy z tabeli `Shipping_methods` i wyświetla je w tabeli HTML.

   - Dla każdej metody dostawy wyświetlane są jej ID, nazwa oraz koszt. Dodatkowo dostępne są przyciski do edytowania (`edit_shipping_method.php`) i usuwania (`delete_shipping_method.php`) danej metody.

4. **Obsługa sytuacji, gdy nie ma metod dostawy:**

   - Jeśli w tabeli `Shipping_methods` nie ma żadnych danych, wyświetlany jest komunikat informujący o braku metod dostawy.

Uwagi:

- Skrypt wymaga, aby użytkownik miał rolę `admin`, aby mógł zobaczyć stronę i dokonać zmian.

- Formularz pozwala na dodanie nowej metody dostawy poprzez podanie jej nazwy i kosztu.

- Skrypt zawiera przyciski do edycji oraz usuwania metod dostawy.

- Po usunięciu lub edytowaniu metody dostawy użytkownik jest przekierowywany na stronę zarządzania dostawami, aby odświeżyć widok.

Dokumentacja do pliku edit_address.php

Cel pliku:

Plik `edit_address.php` umożliwia użytkownikowi edytowanie danych adresowych, takich jak imię, nazwisko, ulica, miasto, kod pocztowy i numer telefonu. Użytkownik może edytować tylko swoje adresy. Dostęp do tego pliku mają tylko użytkownicy, którzy są zalogowani na swoje konto.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

   - Skrypt rozpoczyna sesję za pomocą `session_start()`.

   - Następnie sprawdzane jest, czy użytkownik jest zalogowany, sprawdzając zmienną `$_SESSION['user_id']`. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

2. **Sprawdzanie dostępności adresu:**

   - Jeśli w URL znajduje się parametr `address_id`, skrypt pobiera jego wartość i sprawdza, czy adres o takim ID należy do zalogowanego użytkownika.

   - Wykonywane jest zapytanie SQL, które sprawdza, czy istnieje adres przypisany do użytkownika, który pasuje do podanego ID.

   - Jeśli adres nie istnieje lub nie należy do zalogowanego użytkownika, następuje przekierowanie do strony `platnosci.php`.

3. **Formularz edycji adresu:**

   - Skrypt renderuje formularz, który pozwala na edycję adresu użytkownika. Pola formularza są wstępnie wypełnione danymi adresu pobranymi z bazy danych.

   - Formularz zawiera pola do wprowadzenia imienia, nazwiska, ulicy, miasta, kodu pocztowego oraz numeru telefonu. Pole kodu pocztowego ma dodatkowy atrybut `pattern`, który wymusza format `XX-XXX`.

4. **Aktualizacja adresu:**

   - Po wysłaniu formularza metodą POST, skrypt sprawdza, czy wszystkie pola formularza zostały wypełnione.

   - Jeśli wszystkie wymagane dane zostały podane, skrypt wykonuje zapytanie SQL, aby zaktualizować dane w tabeli `addresses` dla odpowiedniego adresu.

   - Po pomyślnym zaktualizowaniu danych, użytkownik jest przekierowywany na stronę `platnosci.php`.

5. **Bezpieczeństwo:**

   - Skrypt korzysta z funkcji `htmlspecialchars()` w celu zabezpieczenia przed atakami XSS, aby wyświetlać dane użytkownika w formularzu w bezpieczny sposób.

6. **Walidacja danych:**

   - Skrypt wymusza, by wszystkie pola były wypełnione. Dodatkowo pole kodu pocztowego sprawdza, czy wartość jest zgodna z wzorcem `\d{2}-\d{3}` (typowy format kodu pocztowego w Polsce).

Uwagi:

- Skrypt zapewnia, że tylko użytkownik, który dodał dany adres, może go edytować.

- Po edytowaniu adresu użytkownik zostaje przekierowany do strony płatności (`platnosci.php`).

- Formularz zawiera podstawową walidację po stronie klienta (np. dla kodu pocztowego).

Dokumentacja do pliku edit_order.php

Cel pliku:

Plik `edit_order.php` umożliwia administratorowi lub pracownikowi sklepu edytowanie statusu zamówienia. Tylko użytkownicy z odpowiednimi uprawnieniami (administrator lub pracownik) mają dostęp do tego skryptu.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

   - Skrypt zaczyna od rozpoczęcia sesji za pomocą `session_start()`.

   - Następnie sprawdzane jest, czy użytkownik jest zalogowany poprzez obecność zmiennej `$_SESSION['user_id']`. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

- Sprawdzane jest również, czy użytkownik ma odpowiedni poziom uprawnień. Dostęp mają tylko użytkownicy o roli 'admin' lub 'pracownik'. Jeśli użytkownik nie spełnia tych warunków, zostaje przekierowany do strony logowania.

2. **Pobieranie zamówienia z bazy danych:**

- Skrypt sprawdza, czy w URL znajduje się parametr `id`, który określa ID zamówienia.

- Na podstawie tego ID wykonywane jest zapytanie SQL, które pobiera szczegóły zamówienia z tabeli `orders`. Jeśli zamówienie o podanym ID nie istnieje, wyświetlany jest komunikat "Zamówienie nie istnieje" i skrypt kończy działanie.

3. **Formularz edycji statusu:**

- Skrypt wyświetla formularz z aktualnymi danymi zamówienia, w tym ID zamówienia, datą zamówienia oraz jego bieżącym statusem.

- Formularz zawiera rozwijane menu (select), w którym użytkownik może wybrać nowy status zamówienia. Dostępne statusy to: `Oczekujące`, `W realizacji`, `Zrealizowane`, `Anulowane`.

- Aktualnie wybrany status zamówienia jest zaznaczony w polu select.

4. **Aktualizacja statusu zamówienia:**

- Po wysłaniu formularza metodą POST, skrypt pobiera nowy status z formularza i wykonuje zapytanie SQL, które aktualizuje status zamówienia w tabeli `orders`.

- Po pomyślnym zaktualizowaniu statusu, użytkownik zostaje przekierowany na stronę `admin.php`, gdzie może zobaczyć listę zamówień.

5. **Bezpieczeństwo:**

- Skrypt korzysta z funkcji `htmlspecialchars()`, aby zabezpieczyć dane przed atakami XSS, szczególnie podczas wyświetlania danych użytkownika w formularzu.

- Dodatkowo, w formularzu `select` dodano atrybut `required`, aby upewnić się, że użytkownik wybierze nowy status zamówienia.

6. **Interfejs użytkownika:**

- Skrypt wykorzystuje framework CSS Bootstrap, aby nadać formularzowi elegancki wygląd i responsywność.

- Po zapisaniu zmian, użytkownik ma możliwość powrotu do listy zamówień klikając przycisk "Powrót do listy zamówień".

Uwagi:

- Skrypt zapewnia, że tylko administratorzy i pracownicy sklepu mogą edytować statusy zamówień.

- Po zapisaniu zmiany statusu, użytkownik zostaje przekierowany na stronę administracyjną (`admin.php`), gdzie znajduje się lista wszystkich zamówień.

Dokumentacja do pliku edit_page.php

Cel pliku:

Plik `edit_page.php` umożliwia administratorowi edytowanie podstron w systemie. Użytkownik o roli administratora może zmieniać tytuł oraz treść podstrony na podstawie jej ID.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

   - Na początku skryptu wykonywana jest funkcja `session_start()`, która umożliwia dostęp do zmiennych sesyjnych.

   - Skrypt sprawdza, czy użytkownik jest zalogowany, sprawdzając obecność zmiennej `$_SESSION['user_id']`. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

   - Następnie skrypt sprawdza, czy użytkownik jest administratorem. Jeżeli tak, ustawia zmienną `$isAdmin` na `true`. Tylko administratorzy mają dostęp do edycji podstron.

2. **Pobieranie danych podstrony:**

   - Jeśli użytkownik jest administratorem, skrypt sprawdza, czy w URL znajduje się parametr `id`, który wskazuje na ID podstrony do edycji.

   - Na podstawie tego ID, wykonywane jest zapytanie SQL, które pobiera dane podstrony (tytuł i treść) z tabeli `podstrony`.

   - Jeśli podstrona istnieje, jej dane są wyświetlane w formularzu. W przeciwnym przypadku skrypt kończy działanie.

3. **Formularz edycji podstrony:**

   - Formularz zawiera dwa pola: jedno do edycji tytułu podstrony oraz drugie do edycji treści podstrony. Pola te są wypełnione danymi pobranymi z bazy danych.

   - Formularz jest obsługiwany metodą `POST`. Po wysłaniu formularza, skrypt pobiera nowe wartości tytułu i treści oraz wykonuje zapytanie SQL aktualizujące te dane w tabeli `podstrony`.

4. **Aktualizacja podstrony w bazie danych:**

   - Po wysłaniu formularza z nowymi danymi, skrypt aktualizuje rekord w tabeli `podstrony`, zmieniając tytuł oraz treść podstrony.

   - Po pomyślnym zapisaniu zmian, użytkownik jest przekierowywany na stronę `podstrony.php`, gdzie mogą być wyświetlane wszystkie podstrony.

5. **Bezpieczeństwo:**

   - Funkcja `htmlspecialchars()` jest używana do zabezpieczenia przed atakami XSS, co pozwala na bezpieczne wyświetlanie danych, takich jak tytuł i treść podstrony, w formularzu.

   - Skrypt zapewnia, że tylko administratorzy mają dostęp do tej funkcji edytowania podstron.

6. **Interfejs użytkownika:**

   - Formularz edycji jest oparty na frameworku Bootstrap, co sprawia, że jest on responsywny i estetyczny.

   - Po zapisaniu zmian, użytkownik zostaje przekierowany do listy podstron (`podstrony.php`).

   - Na stronie wyświetlany jest także stopka z informacją o prawach autorskich.

Uwagi:

- Skrypt wykorzystuje identyfikator podstrony (`id`) do pobrania odpowiednich danych z bazy danych i wyświetlenia formularza edycji.

- Przekierowanie na stronę `podstrony.php` następuje po zapisaniu zmian, co zapewnia użytkownikowi łatwy dostęp do listy podstron po dokonaniu edycji.

Dokumentacja do pliku edit_product.php

Cel pliku:

Plik `edit_product.php` umożliwia edytowanie danych produktu w sklepie internetowym przez administratora lub pracownika. Administrator może zmieniać dane produktu, takie jak nazwa, cena, kategoria, opis oraz zarządzać zdjęciami produktu.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

- Na początku skryptu wykonywana jest funkcja `session_start()`, umożliwiająca dostęp do zmiennych sesyjnych.

- Sprawdzane jest, czy użytkownik jest zalogowany. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

- Sprawdzane jest, czy użytkownik ma odpowiednie uprawnienia (czy jest administratorem lub pracownikiem), aby móc edytować produkty.

2. **Pobieranie danych produktu:**

- Jeśli użytkownik ma odpowiednie uprawnienia, skrypt sprawdza, czy w URL znajduje się parametr `id`, który wskazuje na ID produktu do edycji.

- Na podstawie tego ID wykonywane jest zapytanie SQL, które pobiera dane produktu (nazwa, cena, opis, kategoria) z tabeli `products` oraz informacje o kategorii z tabeli `categories`.

- Jeśli produkt nie istnieje w bazie danych, skrypt kończy działanie.

3. **Formularz edycji produktu:**

- Formularz umożliwia edytowanie następujących danych produktu: nazwa, cena, opis, kategoria (w postaci listy rozwijanej z kategoriami).

- Formularz umożliwia również dodanie nowych zdjęć oraz usunięcie istniejących zdjęć produktu. Użytkownik może wybrać zdjęcia do usunięcia poprzez zaznaczenie checkboxów.

4. **Zarządzanie zdjęciami:**

- Po zapisaniu formularza, jeśli zostały dodane nowe zdjęcia, skrypt zapisuje je na serwerze w katalogu `zdjecia_opis/` i zapisuje ścieżki do zdjęć w tabeli `zdjecia_opis` w bazie danych.

- Jeśli użytkownik zaznaczył zdjęcia do usunięcia, skrypt usuwa pliki z serwera oraz usuwa odpowiednie rekordy z tabeli `zdjecia_opis`.

5. **Aktualizacja danych produktu w bazie danych:**

- Po przesłaniu formularza, skrypt wykonuje zapytanie SQL, aby zaktualizować dane produktu w tabeli `products` (nazwa, cena, opis, kategoria).

6. **Bezpieczeństwo:**

- Funkcja `htmlspecialchars()` jest używana, aby zabezpieczyć przed atakami XSS, zapewniając bezpieczne wyświetlanie danych w formularzu (np. nazwa, cena, opis).

- Skrypt wykorzystuje przygotowane zapytania SQL (prepared statements), co pomaga zapobiec atakom SQL injection.

7. **Interfejs użytkownika:**

   - Formularz edycji wykorzystuje framework Bootstrap, co zapewnia responsywność i estetyczny wygląd.

   - Pod formularzem wyświetlana jest galeria zdjęć produktu z możliwością usuwania zdjęć.

   - Po zapisaniu zmian, użytkownik jest przekierowywany do strony `admin.php`, gdzie może zarządzać produktami.

   - Skrypt umożliwia dodanie nowych zdjęć poprzez wybór plików w formularzu.

Uwagi:

- Skrypt obsługuje zarówno edycję danych produktu, jak i zarządzanie zdjęciami. Administrator może zmieniać dane produktu oraz kontrolować galerię zdjęć przypisaną do produktu.

- Po dokonaniu edycji, skrypt przekierowuje użytkownika z powrotem na stronę administratora, gdzie może zobaczyć zmiany.

- W przypadku dodawania nowych zdjęć, pliki muszą być przesyłane z formularza i zapisywane na serwerze.

- Usuwanie zdjęć odbywa się poprzez zaznaczenie checkboxów, a zdjęcia są usuwane z serwera i bazy danych.

Dokumentacja do pliku edit_shipping_method.php

Cel pliku:

Plik `edit_shipping_method.php` umożliwia edytowanie danych dotyczących sposobu dostawy w sklepie internetowym. Tylko użytkownicy o odpowiednich uprawnieniach (administratorzy) mogą zmieniać dane dotyczące metod dostawy.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

   - Na początku skryptu wykonywana jest funkcja `session_start()`, umożliwiająca dostęp do zmiennych sesyjnych.

   - Sprawdzane jest, czy użytkownik jest zalogowany. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

   - Sprawdzane jest, czy użytkownik ma odpowiednie uprawnienia (czy jest administratorem). Tylko administratorzy mają dostęp do edytowania metod dostawy.

2. **Pobieranie danych metody dostawy:**

   - Jeśli użytkownik jest administratorem, skrypt sprawdza, czy w URL znajduje się parametr `id`, który wskazuje na ID metody dostawy do edycji.

   - Na podstawie tego ID wykonywane jest zapytanie SQL, które pobiera dane metody dostawy (nazwa, koszt) z tabeli `Shipping_methods`.

   - Jeśli metoda dostawy o podanym ID nie istnieje w bazie danych, skrypt kończy działanie.

3. **Formularz edycji metody dostawy:**

   - Formularz umożliwia edytowanie następujących danych metody dostawy: nazwa, koszt.

   - Formularz wyświetla aktualne dane metody dostawy, które mogą być edytowane przez administratora.

   - Użytkownik może wprowadzić nową nazwę metody dostawy oraz zaktualizować koszt dostawy.

4. **Aktualizacja danych w bazie danych:**

   - Po przesłaniu formularza, skrypt wykonuje zapytanie SQL, aby zaktualizować dane metody dostawy w tabeli `Shipping_methods` (nazwa, koszt).

5. **Bezpieczeństwo:**

   - Funkcja `htmlspecialchars()` jest używana, aby zabezpieczyć przed atakami XSS, zapewniając bezpieczne wyświetlanie danych w formularzu (np. nazwa, koszt).

   - Skrypt wykorzystuje przygotowane zapytania SQL (prepared statements), co pomaga zapobiec atakom SQL injection.

6. **Interfejs użytkownika:**

   - Formularz edycji jest stylizowany za pomocą frameworka Bootstrap, co zapewnia responsywność i estetyczny wygląd.

   - Pod formularzem znajduje się przycisk „Zapisz zmiany” oraz przycisk „Powrót” do strony z listą metod dostawy.

7. **Przekierowanie po zapisaniu zmian:**

   - Po zapisaniu zmian, skrypt przekierowuje użytkownika na stronę `shipping_methods.php`, gdzie może zobaczyć zaktualizowaną listę metod dostawy.

Uwagi:

- Skrypt umożliwia edytowanie danych metody dostawy tylko administratorom.

- Skrypt korzysta z mechanizmu sesji do zarządzania dostępem do strony, weryfikując, czy użytkownik jest administratorem.

- Po dokonaniu edycji, użytkownik jest przekierowywany do strony, gdzie może zobaczyć zmiany.

Dokumentacja do pliku edit_users.php

Cel pliku:

Plik `edit_users.php` umożliwia edytowanie danych użytkowników w panelu administracyjnym sklepu internetowego. Tylko użytkownicy posiadający odpowiednie uprawnienia (administratorzy lub pracownicy) mogą modyfikować dane innych użytkowników.

Działanie pliku:

1. **Sprawdzanie sesji użytkownika:**

   - Na początku skryptu wykonywana jest funkcja `session_start()`, aby umożliwić dostęp do zmiennych sesyjnych.

   - Sprawdzane jest, czy użytkownik jest zalogowany. Jeśli użytkownik nie jest zalogowany, następuje przekierowanie do strony logowania (`login.php`).

   - Dodatkowo, sprawdzane jest, czy użytkownik ma odpowiednie uprawnienia (czy jest administratorem lub pracownikiem). Tylko użytkownicy z odpowiednimi uprawnieniami mogą edytować dane innych użytkowników.

2. **Pobieranie danych użytkownika do edycji:**

   - Skrypt sprawdza, czy w URL znajduje się parametr `id`, który wskazuje na ID użytkownika do edytowania.

   - Na podstawie tego ID wykonywane jest zapytanie SQL, które pobiera dane użytkownika (nazwa użytkownika, e-mail, rodzaj konta) z tabeli `users`.

   - Jeśli użytkownik o podanym ID nie istnieje, skrypt kończy działanie i wyświetla komunikat "Użytkownik nie znaleziony."

3. **Formularz edycji użytkownika:**

   - Formularz umożliwia edytowanie danych użytkownika, takich jak: nazwa użytkownika, e-mail oraz rodzaj konta.

   - Formularz wyświetla aktualne dane użytkownika, które mogą być edytowane przez administratora lub pracownika.

- W formularzu dostępna jest lista dostępnych rodzajów kont (np. `admin`, `pracownik`, `klient`), którą użytkownik może wybrać.

4. **Aktualizacja danych użytkownika:**

- Po przesłaniu formularza, skrypt wykonuje zapytanie SQL, aby zaktualizować dane użytkownika w tabeli `users` (nazwa użytkownika, e-mail, rodzaj konta).

5. **Bezpieczeństwo:**

- Funkcja `htmlspecialchars()` jest używana, aby zabezpieczyć przed atakami XSS, zapewniając bezpieczne wyświetlanie danych w formularzu (np. nazwa użytkownika, e-mail).

- Skrypt korzysta z przygotowanych zapytań SQL (prepared statements), co pomaga zapobiec atakom SQL injection.

6. **Interfejs użytkownika:**

- Formularz edycji jest stylizowany za pomocą frameworka Bootstrap, co zapewnia responsywność i estetyczny wygląd.

- Pod formularzem znajduje się przycisk „Zapisz zmiany” oraz przycisk „Anuluj”, który przekierowuje użytkownika z powrotem do listy użytkowników (`uzytkownicy.php`).

7. **Przekierowanie po zapisaniu zmian:**

- Po zapisaniu zmian skrypt przekierowuje użytkownika na stronę `uzytkownicy.php`, gdzie mogą oni zobaczyć zaktualizowaną listę użytkowników.

Uwagi:

- Skrypt umożliwia edytowanie danych użytkowników tylko użytkownikom, którzy posiadają odpowiednie uprawnienia (administratorom i pracownikom).

- Po zapisaniu zmian, użytkownik jest przekierowywany na stronę, na której może zobaczyć zaktualizowaną listę użytkowników.

Dokumentacja do pliku get_products.php

Cel pliku:

Plik `get_products.php` jest używany do pobierania danych o produktach z bazy danych w formacie JSON. Skrypt umożliwia filtrację produktów według kategorii (jeśli parametr `type` jest przekazany w zapytaniu GET).

Działanie pliku:

1. **Połączenie z bazą danych:**

   - Skrypt rozpoczyna się od nawiązania połączenia z bazą danych MySQL przy użyciu PDO. Używane są dane do połączenia, takie jak host, nazwa bazy danych, użytkownik i hasło, które są zawarte w pliku `db.php`.

   - Jeśli połączenie nie powiedzie się, skrypt zwróci błąd w formacie JSON, zawierający wiadomość o błędzie.

2. **Pobieranie parametru `type`:**

   - Skrypt sprawdza, czy w URL znajduje się parametr GET o nazwie `type`, który określa typ kategorii produktów, które mają zostać pobrane.

   - Jeśli parametr `type` nie jest przekazany, skrypt pobierze wszystkie produkty z bazy danych.

3. **Zapytanie SQL:**

   - Jeśli parametr `type` jest pusty (nie przekazano go w URL), skrypt wykonuje zapytanie SQL, które pobiera wszystkie produkty wraz z kategorią, nazwą, ceną i obrazem.

   - Jeśli parametr `type` jest przekazany, zapytanie SQL jest modyfikowane tak, aby pobrać tylko produkty z określoną kategorią. Kategoria jest porównywana w sposób nieczuły na wielkość liter (`LOWER(c.type)`).

4. **Zwracanie wyników:**

   - Skrypt wykonuje zapytanie SQL i pobiera wyniki, które są następnie zwracane w formacie JSON. Każdy produkt zawiera: `id`, `name`, `price`, `image` oraz `category_type`.

   - Jeśli zapytanie SQL zakończy się błędem, skrypt zwróci błąd w formacie JSON, zawierający szczegóły o błędzie.

5. **Błędy:**

   - Skrypt obejmuje mechanizm obsługi wyjątków, który łapie błędy związane z połączeniem do bazy danych oraz błędy zapytania SQL.

   - W przypadku błędu połączenia lub zapytania, skrypt zwraca szczegóły błędu w formacie JSON.

Interfejs:

- Skrypt nie wyświetla danych w interfejsie użytkownika, tylko zwraca je w formacie JSON.

- JSON zawiera listę produktów z ich podstawowymi danymi: ID, nazwą, ceną, obrazkiem i typem kategorii.

Przykład zapytania GET:

- Aby pobrać wszystkie produkty:

- Aby pobrać produkty z określoną kategorią, np. "akcesoria":

Przykład odpowiedzi JSON (bez filtracji):

```json
[
  {
    "id": 1,
    "name": "Motocykl A",
    "price": 5000,
    "image": "motocykl_a.jpg",
    "category_type": "akcesoria"
  },
  {
    "id": 2,
    "name": "Motocykl B",
    "price": 6000,
    "image": "motocykl_b.jpg",
    "category_type": "motocykle"
  }
]
```

Dokumentacja do pliku index.php

1. Sprawdzenie, czy użytkownik jest zalogowany:

   - Na początku kod sprawdza, czy użytkownik jest zalogowany przy pomocy zmiennej $loggedIn. Jeśli nie, wyświetlane są tylko dostępne opcje bez możliwości zarządzania zamówieniami czy produktami.

2. Wyświetlanie aktualnych zamówień:

- Jeśli użytkownik jest zalogowany i ma jakiekolwiek aktywne zamówienia (pobrane z bazy danych), te zamówienia są wyświetlane w sekcji "Aktualne zamówienia".

- Każde zamówienie pokazuje szczegóły, takie jak ID zamówienia, data, status oraz lista zamówionych produktów (nazwa, ilość i cena).

- Jeśli nie ma produktów w zamówieniu, wyświetlany jest komunikat "Brak produktów w zamówieniu".

3. Wyświetlanie zakończonych zamówień:

- Podobnie jak w przypadku zamówień aktualnych, wyświetlane są zakończone zamówienia, jeśli istnieją w bazie danych.

- Jeśli brak zakończonych zamówień, wyświetlany jest komunikat "Brak zakończonych zamówień".

4. Panel Administracyjny:

- Jeśli użytkownik ma rolę 'admin', dostępny jest panel administracyjny z linkami do zarządzania użytkownikami, produktami, zamówieniami, kategoriami, podstronami i dostawami.

5. Panel Pracownika:

- Jeśli użytkownik ma rolę 'pracownik', dostępny jest panel z opcjami zarządzania produktami, zamówieniami i kategoriami.

6. Modal z danymi użytkownika:

- Zawiera formularz umożliwiający zmianę danych konta (nazwa użytkownika, hasło, e-mail).

- Hasło musi mieć co najmniej 8 znaków, a nowe hasło musi być zgodne z potwierdzeniem.

7. Modal do dodania adresu:

- Formularz umożliwiający użytkownikowi dodanie nowego adresu (imię, nazwisko, ulica, miasto, kod pocztowy, numer telefonu).

- Adres e-mail jest wymagany w formacie "XX-XXX" (np. 00-123), a numer telefonu w formacie np. "+48 123 456 789".

- Po poprawnym wypełnieniu formularza adres jest dodawany do bazy danych.

8. Walidacja formularzy:

- Formularz zmiany danych użytkownika sprawdza, czy nowe hasło spełnia wymogi długości i czy oba hasła się zgadzają.

- Formularz dodawania adresu sprawdza poprawność formatu kodu pocztowego oraz numeru telefonu.

9. Funkcje JavaScript:

- `toggleAccountMenu`: Przełącza widoczność menu konta użytkownika.

- `openadresModal` i `closeadresModal`: Otwiera i zamyka modal do dodawania adresu.

- `openpasswordModal` i `closepasswordModal`: Otwiera i zamyka modal z danymi użytkownika.

- `validateAddressForm`: Sprawdza, czy formularz dodawania adresu zawiera poprawne dane (kod pocztowy, numer telefonu).

10. Obsługa kliknięć poza modalem:

- Jeśli użytkownik kliknie poza modalem (np. na tle), modal zostanie zamknięty.

11. Stylizacja i responsywność:

- Użyto CSS, aby formularze, modale i przyciski były dobrze widoczne i odpowiednio wyświetlały się na różnych rozdzielczościach ekranów.

Dzięki tej dokumentacji, każdy element kodu powinien być zrozumiały w kontekście jego działania w aplikacji.

Dokumentacja do pliku: kategorie.php

1. Sprawdzenie, czy użytkownik jest zalogowany:

- Na początku kod sprawdza, czy użytkownik jest zalogowany przy pomocy zmiennej $_SESSION['user_id']. Jeśli użytkownik nie jest zalogowany, wyświetla komunikat "Zaloguj się, aby edytować kategorie." i kończy wykonanie skryptu.

2. Sprawdzenie uprawnień użytkownika:

- Jeśli użytkownik jest zalogowany, pobierane są jego dane (id i rodzaj - "admin" lub "pracownik") z bazy danych.

- Jeśli użytkownik nie ma odpowiednich uprawnień (nie jest administratorem ani pracownikiem), wyświetlany jest komunikat "Brak dostępu. Tylko administratorzy i pracownicy mogą edytować kategorie." i skrypt kończy działanie.

3. Połączenie z bazą danych:

- Skrypt łączy się z bazą danych za pomocą PDO, korzystając z danych zdefiniowanych w zmiennych $host, $db, $user i $pass.

- Jeśli połączenie nie uda się, wyświetlany jest błąd.

4. Dodawanie i edytowanie kategorii:

- Jeśli metoda żądania to POST, skrypt sprawdza, czy została przesłana nazwa kategorii (type).

- Jeśli nazwa kategorii jest pusta, wyświetlany jest komunikat o błędzie.

- Jeśli identyfikator kategorii (id) jest przekazany, następuje aktualizacja danych w bazie danych. W przeciwnym przypadku, dodawana jest nowa kategoria.

5. Edytowanie istniejącej kategorii:

- Jeśli parametr "edit" jest przekazany w URL, skrypt pobiera dane kategorii do edycji z bazy danych, aby umożliwić użytkownikowi edycję istniejącej kategorii.

6. Wyświetlanie formularza:

- W formularzu wyświetlana jest nazwa kategorii (jeśli kategoria jest edytowana), a przycisk zmienia się na "Zaktualizuj" lub "Dodaj", w zależności od kontekstu.

7. Wyświetlanie istniejących kategorii:

- Skrypt pobiera wszystkie kategorie z bazy danych i wyświetla je w tabeli. Dla każdej kategorii dostępne są opcje edycji oraz usunięcia.

- Linki do edytowania i usuwania kategorii są generowane dynamicznie na podstawie id kategorii.

8. Usuwanie kategorii:

- Link do usunięcia kategorii przekazuje parametr 'id' do skryptu `delete_category.php`. Przed wykonaniem usunięcia użytkownik jest proszony o potwierdzenie akcji.

9. Nawigacja:

- W górnej części strony znajdują się linki umożliwiające przejście do innych sekcji panelu administracyjnego (np. zarządzanie użytkownikami, produktami, zamówieniami).

10. Stylistyka:

- Strona korzysta z frameworku Bootstrap do stylizacji formularzy, przycisków, tabel i innych elementów interfejsu.

- Dodatkowo zastosowano stylowanie tabeli, aby poprawić widoczność tekstu w ciemnym tle.

11. Footer:

- Na dole strony znajduje się stopka z informacją o prawach autorskich sklepu.

Dzięki tej dokumentacji, każdy element kodu będzie jasny i przejrzysty dla osoby, która będzie z niego korzystać lub go modyfikować.

Dokumentacja do pliku: login.php

1. Rozpoczęcie sesji i połączenie z bazą danych:

- Na początku skryptu uruchamiana jest sesja za pomocą `session_start()`, a następnie nawiązywane jest połączenie z bazą danych za pomocą obiektu `mysqli`.

- Jeśli połączenie z bazą danych nie powiedzie się, wyświetlany jest błąd.

2. Przetwarzanie formularza logowania:

- Jeśli użytkownik wypełni formularz i kliknie przycisk "Zaloguj się", kod wykonuje zapytanie do bazy danych, aby sprawdzić, czy istnieje użytkownik o podanej nazwie użytkownika.

- Jeśli użytkownik istnieje, porównywana jest wartość hasła wprowadzonego przez użytkownika z hashem zapisanym w bazie danych (funkcja `password_verify()`).

- Jeśli hasło jest poprawne, sesja jest tworzona z ID i nazwą użytkownika, a użytkownik jest przekierowywany na stronę główną lub stronę administratora, jeśli jego nazwa użytkownika to "admin1234".

- W przypadku nieprawidłowego hasła lub braku użytkownika, wyświetlany jest odpowiedni komunikat o błędzie.

3. Funkcjonalność przypomnienia hasła:

- Skrypt obsługuje również możliwość przypomnienia hasła.

- Jeśli użytkownik kliknie w link "Nie pamiętam hasła", wówczas zostaje wyświetlony formularz do wprowadzenia adresu e-mail.

- Po wypełnieniu formularza, skrypt sprawdza, czy e-mail istnieje w bazie danych.

- Jeśli e-mail jest poprawny, wyświetlany jest komunikat o wysłaniu linku resetującego hasło.

- W przypadku, gdy e-mail nie istnieje w bazie, wyświetlany jest komunikat o błędzie.

4. Formularz logowania:

- Formularz logowania zawiera dwa pola: "Nazwa użytkownika" oraz "Hasło", które są wymagane.

- Po wysłaniu formularza wykonywane są odpowiednie operacje, aby zweryfikować dane użytkownika.

5. Formularz resetowania hasła:

- Po kliknięciu w link "Nie pamiętam hasła" wyświetla się formularz do wprowadzenia adresu e-mail. Po jego wypełnieniu wysyłany jest link do resetu hasła (funkcjonalność wymaga konfiguracji wysyłania e-maili).

6. Nawigacja:

- W dolnej części strony znajdują się dwa przyciski, które prowadzą do stron rejestracji oraz strony głównej.

- Formularz przypomnienia hasła jest ukryty domyślnie i pojawia się dopiero po kliknięciu w odpowiedni link.

7. Style:

- Strona korzysta z frameworku Bootstrap do stylizacji formularzy, przycisków i innych elementów interfejsu.

- Dodatkowo, używany jest zewnętrzny plik CSS `style_login_rej.css` do dostosowania wyglądu strony logowania.

8. Skrypt JavaScript:

- Po kliknięciu w link "Nie pamiętam hasła", skrypt JavaScript wyświetla formularz resetowania hasła, zmieniając jego styl z `display: none` na `display: block`.

Dzięki tej dokumentacji użytkownik będzie w stanie zrozumieć funkcjonalności pliku `login.php` i w razie potrzeby dostosować kod do swoich wymagań.

Dokumentacja do pliku: logout.php

1. Rozpoczęcie sesji:

   - Na początku skryptu wywoływana jest funkcja `session_start()`, która rozpoczyna sesję użytkownika. Sesja musi być rozpoczęta przed jakimikolwiek operacjami związanymi z danymi sesji.

2. Zakończenie sesji:

   - Funkcja `session_destroy()` służy do zakończenia bieżącej sesji użytkownika. Ta operacja powoduje usunięcie wszystkich danych sesji z serwera, co de facto "wylogowuje" użytkownika.

3. Przekierowanie użytkownika:

   - Po zakończeniu sesji użytkownik jest przekierowywany na stronę logowania (`login.php`), co oznacza, że po wylogowaniu nie będzie mógł uzyskać dostępu do zasobów chronionych, dopóki się ponownie nie zaloguje.

4. Zakończenie działania skryptu:

   - Funkcja `exit` zapewnia, że po przekierowaniu nie będzie już wykonywana żadna dalsza część kodu. Jest to dobre praktyki, aby zakończyć działanie skryptu po wykonaniu najważniejszych operacji, jak np. przekierowanie.

Dzięki temu skryptowi użytkownik może wylogować się z systemu i zostać przekierowany na stronę logowania.

Dokumentacja do pliku: orders.php

1. Rozpoczęcie sesji:

   - Na początku skryptu wywoływana jest funkcja `session_start()`, która rozpoczyna sesję użytkownika. Sesja musi być rozpoczęta przed jakimikolwiek operacjami związanymi z danymi sesji.

2. Sprawdzanie, czy użytkownik jest zalogowany:

- Skrypt sprawdza, czy użytkownik jest zalogowany (czy zmienna `$_SESSION['user_id']` jest ustawiona). Jeśli użytkownik nie jest zalogowany, zostaje przekierowany do strony logowania (`login.php`), a dalsze działanie skryptu jest zatrzymywane poprzez `exit()`.

3. Pobieranie zamówień użytkownika:

- Skrypt przygotowuje zapytanie SQL do pobrania zamówień użytkownika z bazy danych na podstawie jego `user_id`. Zapytanie wykonuje się przy pomocy przygotowanego zapytania SQL (`$pdo->prepare()`), a dane są pobierane za pomocą `fetchAll()` w formacie tablicy asocjacyjnej.

4. Generowanie HTML:

- Skrypt generuje HTML, który wyświetla informacje o zamówieniach użytkownika. Jeśli użytkownik ma zamówienia, wyświetlane są one w tabeli, zawierającej następujące kolumny:
  - ID zamówienia
  - Status zamówienia
  - Data zamówienia (formatowana na `Y-m-d H:i:s`)
  - Kwota zamówienia (formatowana na dwie liczby po przecinku, z separacją tysięcy)
  - Link do szczegółów zamówienia

5. Warunki brzegowe:

- Jeżeli użytkownik nie ma żadnych zamówień, wyświetlany jest komunikat "Nie masz żadnych zamówień."

6. Wyswietlanie przycisków nawigacyjnych:

- Na stronie widoczna jest lista przycisków do nawigacji po panelu administracyjnym, takich jak:
  - Strona główna
  - Zarządzanie użytkownikami
  - Zarządzanie kategoriami
  - Zarządzanie zamówieniami
  - Zarządzanie podstronami
  - Zarządzanie produktami
  - Zarządzanie dostawami

7. Linki do szczegółów zamówienia:

- Każde zamówienie w tabeli ma link "Szczegóły", który prowadzi do strony `order_details.php` z parametrem `order_id` zawierającym ID zamówienia. Skrypt generuje odpowiedni URL.

W skrócie, plik `orders.php` wyświetla użytkownikowi jego zamówienia, a także zapewnia możliwość przejścia do szczegółów wybranego zamówienia.

Dokumentacja do pliku: platnosci.css

1. **Podstawowe style:**

- `background-color` dla `body` ustawiony na ciemny odcień (`#1b1b1b`), zapewniający ciemne tło dla całej strony.

- Kolor tekstu ustawiony na czarny (`#000000`) oraz zastosowanie czcionki `Roboto`, zapewniającej nowoczesny wygląd tekstu.

- `margin` i `padding` są ustawione na 0, co usuwa domyślne marginesy i wypełnienie elementów w dokumencie.

2. **Ustawienia dla wszystkich elementów (`*`):**

- Zastosowanie `box-sizing: border-box;`, co oznacza, że szerokość i wysokość elementów obejmuje również marginesy i obramowania.

3. **Kontener główny (`.container`):**

- Użycie flexboxa (`display: flex`) dla układu, z `justify-content: space-between`, co rozdziela elementy w kontenerze na boki.

- Dodano przestrzeń między elementami za pomocą `gap: 30px`.

- Dodatkowo, kontener ma animację `fadeIn`, która sprawia, że cała strona pojawia się płynnie.

4. **Formularz (`.form-container`):**

- Formularz ma szerokość 60% kontenera, ciemne tło (`#2c2c2c`), zaokrąglone rogi, cień (`box-shadow`), a także animację `slideIn`, która wprowadza go na stronę.

- Nagłówki formularza (`h2`) są wyśrodkowane z marginesem na dole.

- Style dla etykiet formularza (`.form-label`) i pól wejściowych (`input`) z ciemnym tłem i białym tekstem.

- Przycisk formularza (`button`) z przejściem w tle i efektami transformacji na hover (powiększenie i zmiana koloru).

5. **Koszyk (`.cart-box`):**

   - Koszyk ma delikatne tło, cień oraz zaokrąglone rogi. Zawiera także animację `slideInCart`, aby pojawiał się z dołu.

   - Nagłówek koszyka (`h4`) jest wyśrodkowany, z większym rozmiarem czcionki.

   - Lista przedmiotów w koszyku (`.list-group`) jest wyświetlana za pomocą flexboxa, a poszczególne elementy listy (`.list-group-item`) mają animację `fadeInItem`, która sprawia, że pozycje pojawiają się z opóźnieniem.

   - Podsumowanie całkowitej kwoty zamówienia wyświetlane jest w prawym dolnym rogu.

6. **Przyciski w koszyku:**

   - Przycisk do finalizacji zamówienia (`.btn-red`) ma czerwone tło, białą czcionkę i cień. Efekt hover sprawia, że przycisk zmienia kolor i delikatnie się powiększa.

7. **Animacje:**

   - `@keyframes fadeIn`: Płynne pojawianie się kontenera głównego.

   - `@keyframes slideInCart`: Animacja wprowadzająca koszyk na stronę.

   - `@keyframes fadeInItem`: Animacja dla pozycji w koszyku, które pojawiają się kolejno.

   - `@keyframes slideIn`: Animacja wprowadzająca formularz.

8. **Modal:**

   - Modal jest ukryty (`display: none`) i wyświetla się na środku ekranu, z przezroczystym tłem.

   - `.modal-content` zawiera białe tło, zaokrąglone rogi oraz marginesy wokół.

9. **Metody płatności:**

   - Elementy metod płatności (`.payment-methods`) są wyświetlane w wierszu z użyciem `flexbox` i zmieniają układ w kolumnę na mniejszych ekranach (w stylach medialnych).

   - Przycisk metody płatności jest klikalny, z efektem powiększenia oraz zmianą koloru tła podczas hover (`.payment-method:hover`).

   - Zaznaczona metoda płatności ma inne tło i obramowanie.

10. **Media queries:**

    - Dla ekranów o szerokości mniejszej niż 768px, kontener koszyka zmienia rozmiar paddingu, a tekst w koszyku i formularzu jest dostosowywany do mniejszych rozmiarów.

Plik `platnosci.css` jest skonstruowany tak, aby strona miała ciemny, elegancki wygląd z płynnie wchodzącymi elementami oraz efektem hover na przyciskach i metodach płatności. Dzięki responsywności, strona dostosowuje się do mniejszych ekranów.

Dokumentacja do pliku: platnosci.js

1. **Dodanie Event Listenera do przycisku wywołującego modal:**

```javascript
document.getElementById("paymentButton").addEventListener("click", function() {
    document.getElementById("paymentModal").style.display = "block";
});
```

Dokumentacja do pliku: platnosci.php

1. Inicjalizacja i połączenie z bazą danych:

- Skrypt zaczyna się od uruchomienia sesji za pomocą `session_start()`, co umożliwia przechowywanie danych użytkownika, np. w koszyku.

- Dołącza się plik `db.php`, który łączy się z bazą danych.

2. Funkcja `clearCart()` - czyszczenie koszyka:

- Funkcja sprawdza, czy użytkownik jest zalogowany. Jeśli tak, usuwa produkty z koszyka w bazie danych.

- Jeśli użytkownik nie jest zalogowany, koszyk przechowywany w sesji zostaje wyczyszczony.

3. Funkcja `calculateTotalAmount($cartItems)` - obliczanie łącznej kwoty:

- Funkcja przechodzi przez wszystkie produkty w koszyku, mnoży cenę przez ilość i sumuje wartości, aby obliczyć całkowitą kwotę zamówienia.

4. Obsługa formularza płatności:

- Jeśli formularz płatności jest wysłany (metoda POST), kod weryfikuje, czy użytkownik jest zalogowany.

- Dla zalogowanych użytkowników pobierane są dane koszyka i metod dostawy/płatności z bazy danych.

- Dla niezalogowanych użytkowników dane o koszyku są pobierane z sesji.

- Jeśli użytkownik nie jest zalogowany, dodawany jest nowy adres do bazy danych.

- Po zakończeniu procesu zamówienia dane o zamówieniu oraz produktach są zapisywane w bazie.

5. Pobieranie produktów z koszyka i metod płatności/dostawy:

- W zależności od statusu użytkownika (zalogowany/niezalogowany) pobierane są odpowiednie dane z bazy.

- Produkty są pobierane z tabeli `carts` i powiązane z tabelą `products`.

- Metody płatności oraz dostawy są pobierane z tabel `shipping_methods` oraz `payment_methods`.

6. Generowanie formularza płatności:

- Formularz płatności umożliwia użytkownikowi wybór metody dostawy, metody płatności oraz adresu dostawy.

- Jeśli użytkownik ma zapisane adresy, może wybrać jeden z nich, w przeciwnym razie może dodać nowy adres.

- Wyświetlana jest całkowita kwota do zapłaty.

7. Koszyk produktów:

- Na stronie wyświetlany jest widok koszyka, pokazujący nazwę produktu, jego cenę i ilość.

- Na końcu wyświetlana jest całkowita kwota do zapłaty za produkty.

8. Obsługa przekierowania:

- Po zapisaniu zamówienia użytkownik jest przekierowywany do strony `thank_you.php`, która informuje o zakończeniu procesu zakupu.

Dokumentacja do pliku: podstrony.php

1. **Inicjalizacja i połączenie z bazą danych:**

- Skrypt rozpoczyna sesję za pomocą `session_start()`, aby umożliwić przechowywanie danych użytkownika w sesji.

- Dołącza plik `db.php`, który odpowiedzialny jest za połączenie z bazą danych.

2. **Sprawdzanie uprawnień użytkownika:**

- Skrypt sprawdza, czy użytkownik jest zalogowany (`$loggedIn`).

- Jeśli użytkownik jest zalogowany, wykonuje zapytanie do bazy danych w celu sprawdzenia, czy jest administratorem (`rodzaj` w tabeli `users`).

- Jeśli użytkownik nie jest administratorem, zostaje przekierowany na stronę logowania (`login.php`).

3. **Pobieranie danych z bazy:**

- Skrypt pobiera dane o podstronach z tabeli `podstrony` za pomocą zapytania SQL:

```php
$sql = "SELECT * FROM podstrony";
$result = $pdo->query($sql);
```

Dokumentacja do pliku: register.php

1. **Inicjalizacja i połączenie z bazą danych:**

- Skrypt rozpoczyna sesję za pomocą `session_start()`.

- Dołącza plik `db.php` odpowiedzialny za połączenie z bazą danych.

2. **Przetwarzanie formularza rejestracji:**

- Skrypt sprawdza, czy formularz został wysłany za pomocą metody POST i czy przycisk "register" został kliknięty.

- Wartości z formularza (nazwa użytkownika, hasło, potwierdzenie hasła, e-mail) są pobierane za pomocą `$_POST`.

3. **Weryfikacja danych rejestracyjnych:**

- Sprawdzanie, czy wszystkie wymagane pola zostały wypełnione.

- Weryfikacja, czy hasła są zgodne.

- Sprawdzanie, czy hasło ma co najmniej 8 znaków.

- Jeśli weryfikacja nie przejdzie, pojawi się odpowiedni komunikat o błędzie.

4. **Sprawdzanie dostępności nazwy użytkownika:**

- Skrypt wykonuje zapytanie SQL, aby sprawdzić, czy użytkownik o podanej nazwie już istnieje w bazie danych.

- Jeśli użytkownik istnieje, wyświetlany jest komunikat o błędzie, informujący o zajętej nazwie.

5. **Rejestracja użytkownika:**

- Jeśli wszystkie dane są poprawne, hasło jest haszowane za pomocą `password_hash()`.

- Następnie wykonywane jest zapytanie SQL, aby dodać nowego użytkownika do bazy danych (`users`).

6. **Komunikat o sukcesie:**

- Po pomyślnej rejestracji użytkownik otrzymuje komunikat o sukcesie.

- Użytkownik zostaje automatycznie przekierowany na stronę logowania po 2 sekundach.

7. **Interfejs użytkownika:**

- Formularz zawiera pola:

  - Nazwa użytkownika

  - Hasło

  - Potwierdzenie hasła

  - E-mail

- Formularz posiada również odpowiednie komunikaty o błędach lub sukcesie, które wyświetlają się w alertach Bootstrap.

- Pod formularzem znajduje się przycisk "Zarejestruj się" oraz link do strony logowania.

8. **Sprawdzanie poprawności danych w JavaScript:**

- Funkcja `checkPassword()` sprawdza, czy hasło ma co najmniej 8 znaków.

- Funkcja `checkPasswordMatch()` sprawdza, czy hasła są zgodne.

- Funkcja `validateForm()` weryfikuje poprawność danych przed wysłaniem formularza. Jeśli dane są niepoprawne, formularz nie zostanie wysłany.

9. **Bezpieczeństwo:**

- Hasło jest haszowane przed zapisaniem do bazy danych, co zapewnia bezpieczeństwo danych użytkownika.

10. **Zakończenie skryptu:**

- Po rejestracji, użytkownik zostaje przekierowany na stronę logowania.

Dokumentacja do pliku: thank_you.php

1. **Inicjalizacja i połączenie z bazą danych:**

- Skrypt rozpoczyna sesję za pomocą `session_start()` i dołącza plik `db.php`, który zapewnia dostęp do bazy danych.

2. **Sprawdzanie, czy użytkownik jest zalogowany:**

- Skrypt sprawdza, czy użytkownik jest zalogowany, czyli czy w sesji znajduje się identyfikator użytkownika (`user_id`).

- Jeśli użytkownik jest zalogowany, pobierane jest ostatnie zamówienie użytkownika z tabeli `orders` (posortowane malejąco po dacie).

3. **Zapytanie do bazy danych - pobranie zamówienia:**

- Jeśli użytkownik jest zalogowany, zapytanie SQL pobiera szczegóły zamówienia (z tabeli `orders`) związane z użytkownikiem.

- Jeśli użytkownik nie jest zalogowany (nie ma sesji `user_id`), skrypt wykonuje zapytanie SQL, które pobiera ostatnie zamówienie bez przypisanego użytkownika (gdzie `user_id` jest NULL).

4. **Weryfikacja i wyświetlanie szczegółów zamówienia:**

- Jeśli nie znaleziono żadnego zamówienia, wyświetlany jest komunikat: "Nie znaleziono zamówienia".

- Jeśli zamówienie istnieje, wykonywane jest kolejne zapytanie SQL, które pobiera produkty powiązane z zamówieniem (z tabel `order_items` i `products`).

- Szczegóły zamówienia są wyświetlane: numer zamówienia, status zamówienia, data zamówienia.

5. **Wyświetlanie produktów w zamówieniu:**

- Produkty w zamówieniu są wyświetlane na liście z nazwą, ilością i ceną za jednostkę.

- Dla każdego produktu obliczana jest łączna cena (`ilość * cena`) i wyświetlana w formie `Łącznie: [kwota] PLN`.

6. **Całkowita kwota zamówienia:**

- Całkowita kwota zamówienia jest wyświetlana na końcu, korzystając z wartości z kolumny `total_amount` w tabeli `orders`.

7. **Przycisk powrotu do strony głównej:**

- Na dole strony znajduje się przycisk "Wróć do strony głównej", który przekierowuje użytkownika do strony głównej (index.php).

8. **Wykorzystanie Bootstrap do stylizacji:**

- Strona korzysta z frameworka Bootstrap do stylizacji elementów, takich jak formularz, przyciski, lista produktów.

9. **Bezpieczeństwo:**

- Wartości wyświetlane na stronie (np. nazwy produktów, ilości, ceny) są zabezpieczone przed atakami XSS przez funkcję `htmlspecialchars()`.

- Ceny są formatowane do dwóch miejsc po przecinku przy pomocy funkcji `number_format()`.

10. **Stopka:**

- Na stronie znajduje się stopka z rokiem i nazwą sklepu: "Sklep Motocyklowy".

Dokumentacja do pliku: uzytkownicy.php

1. **Inicjalizacja sesji i połączenie z bazą danych:**

- Skrypt rozpoczyna sesję przy pomocy `session_start()` i dołącza plik `db.php`, który zawiera połączenie z bazą danych.

2. **Sprawdzanie uprawnień administratora:**

- Skrypt sprawdza, czy użytkownik jest zalogowany, sprawdzając, czy zmienna `$_SESSION['user_id']` jest ustawiona.

- Jeśli użytkownik jest zalogowany, wykonuje zapytanie SQL, aby pobrać dane użytkownika (w tym rodzaj konta). Jeżeli konto ma rodzaj `admin`, użytkownik ma uprawnienia administratora.

- Jeśli użytkownik nie jest administratorem, następuje przekierowanie do strony logowania (`login.php`).

3. **Pobieranie danych użytkowników:**

- Skrypt wykonuje zapytanie SQL, które pobiera wszystkich użytkowników z tabeli `users` w bazie danych.

- Po uzyskaniu danych o użytkownikach, wyświetlana jest tabela z informacjami o każdym użytkowniku: ID, nazwa użytkownika, email, data utworzenia oraz rodzaj konta.

4. **Wyświetlanie tabeli użytkowników:**

- Jeśli w bazie danych są użytkownicy, dla każdego z nich wyświetlane są odpowiednie dane w tabeli HTML.

- Dla każdego użytkownika dostępne są opcje: edytowania (`edit_users.php?id=...`) oraz usuwania (`delete_users.php?id=...`).

5. **Przycisk powrotu do strony głównej:**

- Na górze strony znajdują się przyciski, które umożliwiają przejście do różnych sekcji panelu administracyjnego, takich jak:
  - Strona główna (`index.php`),
  - Zarządzanie użytkownikami (`uzytkownicy.php`),
  - Zarządzanie kategoriami produktów (`kategorie.php`),
  - Zarządzanie zamówieniami (`zamowienia.php`),
  - Zarządzanie podstronami (`podstrony.php`),
  - Zarządzanie produktami (`admin.php`),
  - Zarządzanie dostawami (`dostawy.php`).

6. **Stylizacja za pomocą Bootstrap:**

- Skrypt korzysta z frameworka Bootstrap (wersja 4.5.2) do stylizacji elementów strony, takich jak tabela użytkowników, przyciski oraz nagłówki.

- Kolor tekstu tabeli jest ustawiony na biały poprzez dodanie stylu CSS w sekcji `<style>`.

7. **Bezpieczeństwo:**

- Dane wyświetlane na stronie są zabezpieczone przed atakami XSS za pomocą funkcji `htmlspecialchars()`. Ponadto, użytkownik musi być zalogowany i mieć odpowiedni rodzaj konta, aby mieć dostęp do tego panelu.

8. **Stopka:**

- Na dole strony znajduje się stopka z informacją o prawach autorskich: "Sklep Motocyklowy".

Dokumentacja do pliku: zamowienia.php

1. **Inicjalizacja sesji i połączenie z bazą danych:**

- Skrypt rozpoczyna sesję przy pomocy `session_start()` i dołącza plik `db.php`, który zawiera połączenie z bazą danych.

2. **Sprawdzanie uprawnień użytkownika:**

- Skrypt sprawdza, czy użytkownik jest zalogowany, poprzez sprawdzenie, czy zmienna `$_SESSION['user_id']` jest ustawiona.

- Dodatkowo, użytkownik musi mieć rolę `admin` lub `pracownik`, aby mieć dostęp do tego panelu. W przypadku braku odpowiednich uprawnień, użytkownik zostaje przekierowany do strony logowania (`login.php`).

3. **Pobieranie danych zamówień:**

- Skrypt wykonuje zapytanie SQL, aby pobrać wszystkie zamówienia z tabeli `orders`, posortowane malejąco według daty zamówienia.

- Po uzyskaniu danych o zamówieniach, skrypt przetwarza każde zamówienie, pobierając powiązane dane o użytkowniku, produktach oraz adresie dostawy.

4. **Wyświetlanie zamówień w tabeli:**

- Na stronie wyświetlana jest tabela z listą zamówień. Dla każdego zamówienia przedstawiane są następujące informacje:

  - ID zamówienia,

  - ID użytkownika (z nazwą użytkownika),

  - Data zamówienia,

  - Status zamówienia,

  - Całkowita kwota zamówienia,

- Lista produktów w zamówieniu (nazwa produktu, ilość, cena jednostkowa),

- Adres dostawy (imię, nazwisko, ulica, miasto, kod pocztowy, telefon).

5. **Opcje edycji i usunięcia zamówienia:**

- Dla każdego zamówienia dostępne są opcje:

  - **Edytuj**: Przycisk przekierowujący do strony edycji zamówienia (`edit_order.php?id=...`),

  - **Usuń**: Przycisk usuwający zamówienie (`delete_order.php?id=...`).

6. **Stylizacja za pomocą Bootstrap:**

- Skrypt korzysta z frameworka Bootstrap (wersja 4.5.2) do stylizacji elementów strony, takich jak tabela zamówień, przyciski oraz nagłówki.

- Kolor tekstu tabeli jest ustawiony na biały za pomocą stylu CSS w sekcji `<style>`.

7. **Stopka:**

- Na dole strony znajduje się stopka z informacją o prawach autorskich: "Sklep Motocyklowy".

8. **Zabezpieczenia:**

- Wszystkie dane wyświetlane na stronie są zabezpieczone przed atakami XSS przy pomocy funkcji `htmlspecialchars()`.

- Skrypt sprawdza uprawnienia użytkownika przed wyświetleniem informacji o zamówieniach, zapewniając, że tylko zalogowani użytkownicy z odpowiednią rolą mogą zarządzać zamówieniami.