# process\_registration.php

## Połączenie z bazą danych

$servername = "localhost";

$username = "ushosteu\_wydatki";

$password = "Szkola123!";

$dbname = "ushosteu\_wydatki";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

Kod odpowiada za nawiązanie połączenia z bazą danych MySQL. Jeśli połączenie nie powiedzie się, skrypt zakończy działanie, wyświetlając komunikat o błędzie.

## Przetwarzanie danych z formularza rejestracyjnego

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $imie = $\_POST["imie"];

    $nazwisko = $\_POST["nazwisko"];

    $email = $\_POST["email"];

    $login = $\_POST["login"];

    $haslo = $\_POST["haslo"];

    $haslo\_powtorz = $\_POST["haslo\_powtorz"];

Kod sprawdza, czy żądanie przyszło metodą POST i pobiera dane z formularza rejestracyjnego.

## Sprawdzenie zgodności haseł

if ($haslo != $haslo\_powtorz) {

        die("Passwords do not match");

    }

Jeśli hasła wprowadzone przez użytkownika nie są zgodne, skrypt kończy działanie, wyświetlając komunikat o niezgodności haseł.

## Szyfrowanie hasła, dodanie użytkownika do bazy danych

$hashed\_password = password\_hash($haslo, PASSWORD\_DEFAULT);

    $sql = "INSERT INTO uzytkownik (Imie, Nazwisko, Email, Login, Haslo) VALUES ('$imie', '$nazwisko', '$email', '$login', '$hashed\_password')";

    if ($conn->query($sql) === TRUE) {

        header("Location: welcome.html");

        exit();

    } else {

        echo "Error: " . $sql . "<br>" . $conn->error;

    }

Skrypt generuje skróconą wersję hasła za pomocą funkcji password\_hash(), a następnie dodaje użytkownika do bazy danych. Jeśli operacja zakończy się sukcesem, przekierowuje użytkownika na stronę welcome.html. W przeciwnym razie wyświetla komunikat o błędzie.

# process\_login.php

## Sprawdzenie metody żądania

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $login = $\_POST["login"];

    $haslo = $\_POST["haslo"];

    $sql = "SELECT id, login, haslo FROM uzytkownik WHERE login = '$login'";

    $result = $conn->query($sql);

Kod sprawdza, czy żądanie przyszło metodą POST i pobiera dane z formularza logowania. Następnie sprawdza, czy użytkownik o podanym loginie istnieje w bazie danych.

## Weryfikacja hasła

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $login = $\_POST["login"];

    $haslo = $\_POST["haslo"];

    $sql = "SELECT id, login, haslo FROM uzytkownik WHERE login = '$login'";

    $result = $conn->query($sql);

    if ($result->num\_rows == 1) {

        $row = $result->fetch\_assoc();

        if (password\_verify($haslo, $row["haslo"])) {

            $\_SESSION["user\_id"] = $row["id"];

            $\_SESSION["user\_login"] = $row["login"];

            header("Location: main.php");

            exit();

        } else {

            echo '<script>alert("Nieprawidłowe hasło. Spróbuj jeszcze raz."); window.location.href = "login.html";</script>';

            exit();

        }

    } else {

        echo '<script>alert("Użytkownik o takiej nazwie nie isnieje. Spróbuj jeszcze raz."); window.location.href = "login.html";</script>';

        exit();

    }

Tu sprawdzane jest, czy hasło podane przez użytkownika jest poprawne. Jeśli tak, ustawiane są zmienne sesji, a użytkownik przekierowywany jest na stronę main.php. W przeciwnym razie wyświetlane są komunikaty o błędzie.

## Funkcja pomocnicza do wyświetlania alertów

function function\_alert($message) {

    echo "<script>alert('$message');</script>";

}

Funkcja umożliwia wygodne wyświetlanie alertów w skrypcie.

# main.php

## Sprawdzenie autoryzacji użytkownika

if (isset($\_SESSION["user\_id"])) {

            echo '<div class="logged-in-info">Zalogowany jako: ' . $\_SESSION["user\_login"] . ' (ID użyt.: ' . $\_SESSION["user\_id"] . ')</div>';

        }

        else {

            echo '<div class="unauthorized-message">Nieautoryzowany dostęp, przenoszę na stronę główną...</div>';

            header("Refresh: 3; URL=index.html");

            exit();

        }

Kod sprawdza, czy użytkownik jest zalogowany. Jeśli tak, wyświetla informacje o zalogowanym użytkowniku. W przeciwnym razie wyświetla komunikat o nieautoryzowanym dostępie i przekierowuje na stronę główną po 3 sekundach.

# process\_edit\_profile.php

## Pobranie danych z formularza i sesji

$user\_id = $\_SESSION["user\_id"];

    $imie = $\_POST["imie"];

    $nazwisko = $\_POST["nazwisko"];

    $email = $\_POST["email"];

    $login = $\_POST["login"];

    $haslo = $\_POST["haslo"];

    $haslo\_powtorz = $\_POST["haslo\_powtorz"];

Kod pobiera dane z formularza przesłane metodą POST oraz identyfikator użytkownika z sesji.

## Haszowanie hasła i aktualizacja profilu w bazie

$hashed\_password = password\_hash($haslo, PASSWORD\_DEFAULT);

    $update\_sql = "UPDATE uzytkownik SET Imie='$imie', Nazwisko='$nazwisko', Email='$email', Login='$login', Haslo='$hashed\_password' WHERE id='$user\_id'";

    if ($conn->query($update\_sql) === TRUE) {

        echo '<script>alert("Profile updated successfully."); window.location.href = "profile.php";</script>';

        exit();

    } else {

        echo "Error updating profile: " . $conn->error;

    }

Hasło jest haszowane przy użyciu funkcji password\_hash, a następnie aktualizowane są dane użytkownika w bazie danych. Po pomyślnej aktualizacji, skrypt wyświetla komunikat o sukcesie i przekierowuje użytkownika na stronę profilu. W przypadku błędu aktualizacji, wyświetlany jest komunikat z informacją o błędzie.

## Pobranie danych użytkownika z bazy danych

$user\_id = $\_SESSION["user\_id"];

$select\_sql = "SELECT Imie, Nazwisko, Email, Login FROM uzytkownik WHERE id = '$user\_id'";

        $result = $conn->query($select\_sql);

        if ($result->num\_rows > 0) {

            $row = $result->fetch\_assoc();

            $imie = $row["Imie"];

            $nazwisko = $row["Nazwisko"];

            $email = $row["Email"];

            $login = $row["Login"];

        } else {

            echo "Error fetching user data: User not found";

            exit();

        }

Dane użytkownika są pobierane z bazy danych na podstawie identyfikatora użytkownika z sesji.

# generate\_report.php

## Pobranie istniejących kategorii

$categoriesQuery = "SELECT DISTINCT category FROM expenses WHERE user\_id = {$\_SESSION["user\_id"]}";

$categoriesResult = $conn->query($categoriesQuery);

$existingCategories = [];

while ($categoryRow = $categoriesResult->fetch\_assoc()) {

    $existingCategories[] = $categoryRow["category"];

}

Pobranie unikalnych dla użytkownika kategorii z bazy danych.

## Obsługa parametrów z formularza GET

if ($\_SERVER["REQUEST\_METHOD"] == "GET") {

    if (isset($\_GET["start-date"]) && isset($\_GET["end-date"])) {

        $startDate = date("Y-m-d", strtotime($\_GET["start-date"]));

        $endDate = date("Y-m-d", strtotime($\_GET["end-date"]));

        $selectedCategories = isset($\_GET["category"]) ? $\_GET["category"] : [];

//...

    } else {

        echo '<div class="error-message">Brak wymaganych danych do generowania raportu.</div>';

    }

Sprawdzenie, czy przekazane są wymagane parametry (data początkowa, data końcowa). Pobranie wybranych kategorii z formularza GET.

## Zapytanie do bazy danych ma podstawie wybranych parametrów

$categoryCondition = "";

        if (!empty($selectedCategories)) {

            $categoryCondition = "AND category IN ('" . implode("','", $selectedCategories) . "')";

        }

        $query = "SELECT \* FROM expenses

                  WHERE user\_id = {$\_SESSION["user\_id"]}

                  AND date BETWEEN '$startDate' AND '$endDate'

                  $categoryCondition";

        $result = $conn->query($query);

Dynamiczna konstrukcja zapytania SQL uwzględniającego wybrane parametry (data, kategorie).

## Wyświetlenie wydatków w tabeli html

echo '<table>';

        echo '<tr><th>Data</th><th>Kwota</th><th>Kategoria</th><th>Opis</th></tr>';

        while ($row = $result->fetch\_assoc()) {

            echo "<tr>";

            echo "<td>" . $row["date"] . "</td>";

            echo "<td>" . $row["amount"] . "</td>";

            echo "<td>" . $row["category"] . "</td>";

            echo "<td>" . $row["expense\_name"] . "</td>";

            echo "</tr>";

        }

        echo '</table>';

# logout.php

## Inicjalizacja sesji

session\_start();

## Usunięcie danych sesyjnych i zniszczenie sesji

session\_unset();

session\_destroy();

session\_unset() usuwa wszystkie zarejestrowane zmienne sesyjne,

natomiast session\_destroy() zamyka sesję i usuwa zapisane dane sesyjne.

# table.php

## Budowa zapytania sql dla wydatków z uwzględnieniem wyszukiwania

$sql = "SELECT id, date, category, expense\_name, amount FROM expenses WHERE user\_id = $user\_id";

if (!empty($searchCategory)) {

    $sql .= " AND category LIKE '%$searchCategory%'";

}

if (!empty($searchDate)) {

    $sql .= " AND date = '$searchDate'";

}

if (!empty($searchDescription)) {

    $sql .= " AND expense\_name LIKE '%$searchDescription%'";

}

if (!empty($searchAmount)) {

    $sql .= " AND amount = $searchAmount";

}

Dynamiczna konstrukcja zapytania SQL uwzględniającego filtry wyszukiwania dla kategorii, daty, opisu i kwoty.

## Wykonanie zapytania do bazy danych i wyświetlenie wyników w tabeli html

$result = $conn->query($sql);

//...

if ($result->num\_rows > 0) {

                    while ($row = $result->fetch\_assoc()) {

                        echo "<tr>";

                        echo "<td>" . $row["date"] . "</td>";

                        echo "<td>" . $row["amount"] . "</td>";

                        echo "<td>" . $row["category"] . "</td>";

                        echo "<td>" . $row["expense\_name"] . "</td>";

                        echo "<td><a href='#' onclick='openEditPopup(" . $row["id"] . ", \"" . $row["date"] . "\", \"" . $row["category"] . "\", \"" . $row["expense\_name"] . "\", " . $row["amount"] . ")'><img src='images/edit.png' alt='Edytuj'></a> ";

                        echo "<a href='#' onclick='deleteExpense(" . $row["id"] . ")'><img src='images/delete.png' alt='Usuń'></a></td>";

                        echo "</tr>";

                    }

                } else {

                    echo "<tr><td colspan='5'>Brak wyników wyszukiwania.</td></tr>";

## Funkcja saveExpense

function saveExpense($conn, $date, $category, $expenseName, $amount, $userId) {

    $stmt = $conn->prepare("INSERT INTO expenses (date, category, expense\_name, amount, user\_id) VALUES (?, ?, ?, ?, ?)");

    if (!$stmt) {

        die("Error in SQL query: " . $conn->error);

    }

    $stmt->bind\_param("ssssd", $date, $category, $expenseName, $amount, $userId);

    $stmt->execute();

    $stmt->close();

}

Funkcja saveExpense służy do dodawania nowego wydatku do bazy danych. W przypadku błędu w przygotowaniu zapytania SQL, funkcja kończy się błędem z informacją o błędzie. Funkcja korzysta z prepared statements, co minimalizuje ryzyko ataków SQL injection.

## Funkcja editExpense

function editExpense($conn, $editedDate, $editedCategory, $editedExpenseName, $editedAmount, $editedExpenseId) {

    $stmt = $conn->prepare("UPDATE expenses SET date = ?, category = ?, expense\_name = ?, amount = ? WHERE id = ? AND user\_id = ?");

    if (!$stmt) {

        die("Error in SQL query: " . $conn->error);

    }

    $userId = $\_SESSION["user\_id"];

    $stmt->bind\_param("ssssdd", $editedDate, $editedCategory, $editedExpenseName, $editedAmount, $editedExpenseId, $userId);

    $stmt->execute();

    $stmt->close();

}

Funkcja editExpense umożliwia edytowanie istniejącego wydatku w bazie danych.

## Funkcja deleteExpense

function deleteExpense($conn, $expenseId) {

    $stmt = $conn->prepare("DELETE FROM expenses WHERE id = ? AND user\_id = ?");

    if (!$stmt) {

        die("Error in SQL query: " . $conn->error);

    }

    $userId = $\_SESSION["user\_id"];

    $stmt->bind\_param("dd", $expenseId, $userId);

    $stmt->execute();

    $stmt->close();

}

Funkcja deleteExpense usuwa istniejący wydatek z bazy danych.