Dokumentation Eigenleistung WebEngineering – Login/Logout

Jannik Filpe – 8700390

Einleitung

Das Ziel dieser Implementierung ist es, sicherzustellen, dass sich die Benutzer der Website mit ihren Anmeldedaten authentifizieren müssen, um Zugriff auf die geschützten Bereiche der Seite zu erhalten. Dies geschieht durch die Eingabe einer gültigen Kombination aus Benutzernamen und Passwort auf der Login-Seite. Nur bei erfolgreicher Authentifizierung erhält der Nutzer Zugriff auf sensible Geschäfts- und Personendaten. Bei einer fehlerhaften Eingabe wird der Benutzer darauf hingewiesen und bleibt auf der Login-Seite, um erneute Versuche zu ermöglichen. Diese Maßnahme ist von zentraler Bedeutung, um sicherzustellen, dass Unbefugte keinen Zugriff auf vertrauliche Informationen erhalten.

Einbindung in das Gesamtprojekt

Integration der Login-Funktion:
Die Login-Seite (*login.html*) ist der Einstiegspunkt für alle Benutzer. Diese Seite ist als einzige html-Seite direkt über die URL erreichbar. Sie ist so gestaltet, dass sie sich nahtlos in das Design des gesamten CRM-Systems einfügt. Die Authentifizierung wird serverseitig in der Datei *login.php* durchgeführt, die mit der Datenbank kommuniziert, um die Anmeldedaten zu überprüfen.

Nach erfolgreicher Anmeldung wird eine Session gestartet, die den Benutzer während seiner gesamten Nutzung der Webanwendung authentifiziert. Diese Session ermöglicht es, dass der Benutzer auf alle geschützten Seiten zugreifen kann, ohne sich erneut anmelden zu müssen. Der Zugriff auf geschützte Seiten wie *indexProtected.php*, leadsProtected.php und *customerProtected.php* wird durch eine Überprüfung der aktiven Session ermöglicht. Diese Seiten inkludieren die eigentlichen HTML-Inhalte und sind nur für angemeldete Benutzer zugänglich.

Integration der Logout-Funktion:
Der Logout-Mechanismus ist ebenfalls eng in das CRM-System integriert. Im Dropdown-Menü des Benutzers, das von jeder Seite aus zugänglich ist, befindet sich ein "Abmelden"-Link. Dieser Link verweist auf *logout.php*, ein Skript, das die aktive Session beendet, alle Session-Daten löscht und den Benutzer zurück zur Login-Seite leitet. Dadurch wird sichergestellt, dass nach dem Logout keine vertraulichen Daten mehr zugänglich sind.

Implementierung

1. Frontend: Die Login-Seite

Die Login-Seite (*login.html*) ist der Einstiegspunkt für den Benutzer. Sie besteht aus einem einfachen HTML-Formular, das den Benutzer nach seinen Anmeldedaten (Benutzername und Passwort) fragt. Hier ist der entsprechende Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CRM-Dashboard</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="styles/style.css">
    <script src="scripts/login.js" defer></script>
    <style>
        #error-message {
            color: red;
            font-size: 0.8em;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login</h2>
        <form action="login.php" method="post">
            <input type="text" name="username" placeholder="Benutzername" required>
            <input type="password" name="password" placeholder="Passwort" required>
            <div id="error-message" style="display: none;">
                <p>Falscher Benutzername oder Passwort. Bitte versuche es erneut.</p>
            </div>
            <button type="submit">Anmelden</button>
        </form>
    </div>
</body>
</html>
```

Beschreibung:

- Formularaufbau: Das Formular enthält zwei Eingabefelder für den Benutzernamen und das Passwort sowie eine Schaltfläche zum Absenden.
- Formular-Aktion: Das Formular verwendet die POST-Methode und sendet die eingegebenen Daten an die Datei *login.php*, die auf dem Server ausgeführt wird.

2. Backend: Die Login-Überprüfung

Da es noch keine Möglichkeit auf der Seite z.B. als Admin neue Nutzerkonten zu erstellen, wurde über das Kontrollpanels des Servers manuell ein Eintrag in der Nutzertabelle in der Datenbank eingefügt. Der Nutzername ist „adminUser“ und das Passwort ist „adminPasswort“. In der Datenbank ist das Passwort allerdings als Hashwert gespeichert, um keine Klarpasswörter von Nutzern auf dem Server zu speichern.

Die Datei *login.php* auf dem Server nimmt die gesendeten Formulardaten entgegen und überprüft sie auf Richtigkeit. Hier ist der entsprechende Code:

```php
<?php
session_start();
include 'db_connect.php';


if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = isset($_POST['username']) ? htmlspecialchars($_POST['username']) : '';
    $password = isset($_POST['password']) ? htmlspecialchars($_POST['password']) : '';

    $stmt = $conn->prepare("SELECT password_hash FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $stmt->bind_result($password_hash);
        $stmt->fetch();

        if (password_verify($password, $password_hash)) {
            $_SESSION['loggedin'] = true;
            $_SESSION['username'] = $username;
            header("Location: indexProtected.php");
            exit();
        } else {
            header("Location: login.html?error=1");
            exit();
        }
    } else {
        header("Location: login.html?error=1");
            exit();
    }

    $stmt->close();
    $conn->close();
} else {
    echo "Ungültige Anfragemethode.";
}
?>
```

Beschreibung:

- Session-Verwaltung: Zu Beginn der Datei wird die Session gestartet, um Benutzerdaten zu speichern, falls die Authentifizierung erfolgreich ist.

- Eingabeverarbeitung: Der Server überprüft, ob eine POST-Anfrage vorliegt und verarbeitet dann die übermittelten Anmeldedaten.
- Datenbankverbindung: Eine Verbindung zur Datenbank wird hergestellt, um die Benutzerdaten zu überprüfen.
- SQL-Abfrage: Es wird eine SQL-Abfrage ausgeführt, um den Benutzer mit dem angegebenen Benutzernamen zu finden.
- Passwortüberprüfung: Das eingegebene Passwort wird mit dem in der Datenbank gespeicherten Hash verglichen. Wenn die Daten übereinstimmen, wird die Session aktualisiert und der Benutzer zur geschützten Seite (*indexProtected.php*) weitergeleitet. Andernfalls verbleibt der User auf der Login-Seite und über die URL wird ein error-Paramter mit Wert 1 weitergegeben (siehe 3.)

3. Fehlermeldung bei falschen Login-Daten

Nach fehlerhafter Eingabe von Benutzername und/oder Passwort verbleibt der Nutzer auf der Login-Seite. Allein schon aus Gründen der Nutzerfreundlichkeit und der klaren Gestaltung der Website wird der Nutzer durch eine Fehlermeldung darauf aufmerksam gemacht. Der folgende JavaScript-Code in *login.js* wurde implementiert, um zu prüfen, ob eine Fehlermeldung angezeigt werden muss:

```
function getQueryParam(param) {
    const urlParams = new URLSearchParams(window.location.search);
    return urlParams.get(param);
}

document.addEventListener('DOMContentLoaded', (event) => {
    if (getQueryParam('error') === '1') {
        document.getElementById('error-message').style.display = 'block';
    }
});
```

Beschreibung:

- getQueryParam(param): Diese Funktion extrahiert den Wert eines bestimmten URL-Parameters (in diesem Fall error) aus der aktuellen Seiten-URL. Sie verwendet dazu die URLSearchParams-API, um die URL-Parameter zu durchsuchen und den Wert des gesuchten Parameters zurückzugeben.
- Event Listener (DOMContentLoaded): Diese Funktion wird ausgeführt, sobald das Dokument vollständig geladen ist. Das bedeutet, dass alle DOM-Elemente zur Verfügung stehen.
    - Überprüfung auf den Fehlerparameter: Wenn der error-Parameter in der URL den Wert 1 hat, deutet dies darauf hin, dass ein Login-Fehler aufgetreten ist.
    - Fehlermeldung anzeigen: Wenn der Fehlerparameter gefunden wird, wird das HTML-Element mit der ID error-message sichtbar gemacht, indem dessen display-Eigenschaft auf block gesetzt wird.

4. Schutz der Seiten

Um sicherzustellen, dass nur authentifizierte Benutzer auf die geschützten Seiten zugreifen können, wird eine zusätzliche Überprüfung auf diesen Seiten eingebaut. Zum Beispiel in der *leadsProtected.php*:

```php
<?php
session_start();

if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {
    header("Location: login.html");
    exit();
}

include 'leads.html'
?>
```

Beschreibung:

- Session-Überprüfung: Zu Beginn jeder geschützten Seite wird überprüft, ob die loggedin-Session-Variable gesetzt und true ist. Wenn nicht, wird der Benutzer zur Login-Seite weitergeleitet.

5. Umleitung von HTML auf PHP-Seiten mit *.htaccess*

Um sicherzustellen, dass Benutzer immer auf die geschützten PHP-Seiten zugreifen und nicht auf die ungeschützten HTML-Seiten, wurden entsprechende Umleitungen über *.htaccess* eingerichtet. Außerdem wird mithilfe von *.htaccess* die Startseite auf *indexProtected.php* festgelegt. Die *.htaccess* Datei befindet sich im Stammverzeichnis des Webservers und enthält die folgenden Umleitungen:

```
RewriteEngine On

RewriteRule ^customer\.html$ /customerProtected.php [R=301,L]
RewriteRule ^leads\.html$ /leadsProtected.php [R=301,L]
RewriteRule ^calender\.html$ /calenderProtected.php [R=301,L]
RewriteRule ^customerorders\.html$ /customerordersProtected.php [R=301,L]
RewriteRule ^my_profile\.html$ /my_profileProtected.php [R=301,L]
RewriteRule ^orderquotation\.html$ /orderquotationProtected.php [R=301,L]

DirectoryIndex indexProtected.php

ErrorDocument 403 https://errors.infinityfree.net/errors/403/
ErrorDocument 404 https://errors.infinityfree.net/errors/404/
ErrorDocument 500 https://errors.infinityfree.net/errors/500/
php_value date.timezone Europe/Berlin

<IfModule mod_headers.c>
   Header set Cache-Control "no-cache, no-store, must-revalidate"
   Header set Pragma "no-cache"
   Header set Expires 0
</IfModule>
```

Beschreibung:

- RewriteEngine On: Aktiviert die Rewrite-Engine von Apache, die es ermöglicht, URLs basierend auf bestimmten Regeln umzuschreiben.
- RewriteRule: Jede RewriteRule gibt an, dass eine HTML-Seite (wie *leads.html*) auf die entsprechende PHP-Seite (wie *leadsProtected.php*) umgeleitet wird.
- DirectoryIndex: Setzt die Standardseite für den Fall, dass der Benutzer keine bestimmte Datei in der URL angibt. Hier wird *indexProtected.php* anstelle von *index.html* als Standard festgelegt.

Funktionsweise:

- Wenn ein Benutzer versucht, direkt auf eine der HTML-Seiten zuzugreifen (z. B. durch Eingabe von *crmwebapp.free.nf/leads.html* in die URL-Leiste), wird er automatisch auf die entsprechende geschützte Seite (*leadsProtected.php*) umgeleitet.
- Dies stellt sicher, dass Benutzer die Seite nur nach erfolgreichem Login sehen können, da die Protected-Seiten alle eine Session-Überprüfung durchführen, um sicherzustellen, dass der Benutzer eingeloggt ist.

6. Logout

Der Logout-Mechanismus wurde über eine separate PHP-Datei (*logout.php*) implementiert, die die aktuelle Sitzung beendet und den Benutzer anschließend zur Login-Seite weiterleitet. Der Logout-Button wurde im Dropdown-Menü des Benutzerprofils eingebunden. Der HTML-Code dafür sieht folgendermaßen aus:

```
<div class="dropdown-content" id="profileDropdown">
      <ul>

        <li><a href="my_profile.html">Profil bearbeiten</a></li>
        <li><a href="#">Sicherheitseinstellungen</a></li>
        <li><a href="logout.php">Abmelden</a></li>
      </ul>
  </div>
```

Beim Klicken auf Abmelden wird *logout.php* abgerufen. Das Logout-Skript sorgt dafür, dass die aktuelle Sitzung zerstört und alle zugehörigen Sitzungsdaten gelöscht werden. Hier ist der entsprechende Code:

```
<?php
session_start();

session_unset();

session_destroy();

header("Location: login.html");
exit();
?>
```

Funktionsweise:

- Starten der Sitzung: session_start() stellt sicher, dass die aktuelle Sitzung aktiv ist und auf die Sitzungsvariablen zugegriffen werden kann.
- Entfernen aller Sitzungsvariablen: session_unset() entfernt alle Variablen, die in der aktuellen Sitzung gespeichert sind. Dadurch wird sichergestellt, dass keine sensiblen Daten mehr in der Sitzung verbleiben.
- Zerstören der Sitzung: Mit session_destroy() wird die Sitzung vollständig beendet. Dies verhindert, dass ein Benutzer auf die zuvor gespeicherten Sitzungsdaten zugreifen kann.
- Weiterleitung zur Login-Seite: Nach dem erfolgreichen Beenden der Sitzung wird der Benutzer mit header("Location: *login.html*"); zur Login-Seite weitergeleitet. Diese Weiterleitung stellt sicher, dass der Benutzer erneut authentifiziert werden muss, bevor er auf die geschützten Bereiche der Website zugreifen kann.

Fazit

Die Implementierung des Login- und Logout-Systems für die CRM-Webanwendung gewährleistet den sicheren Zugriff auf geschützte Bereiche der Website. Durch die Verwendung von PHP für serverseitige Authentifizierung und Session-Management, kombiniert mit .htaccess für die URL-Weiterleitung, konnte eine effektive Zugriffskontrolle erreicht werden. Insgesamt sorgt das System

dafür, dass nur authentifizierte Benutzer Zugang zu sensiblen Daten erhalten und alle sicherheitsrelevanten Anforderungen erfüllt werden.