



DOKUMENTATION

HowToWebsite

LAM

LOPES CALDEIRA FILIPE
THIELEN ALEXANDRO
MEKIYEV DENI
WIRTGEN LOU

2GIN1
2024

1. PLANUNG	3
Filipe:.....	3
Register, Forum:	3
Deni:	3
Log-in, User Dashboard, CSS/Styling:	3
Lou:.....	4
Alexandro:	4
2. STRUKTUR	5
Log-in und Register Formular:	5
Register:	5
Login:	5
Main Page:	5
Top Navigation:	5
Information Cards:	5
Contributors:	5
Course Page:	6
Set-up:	6
Display:	6
User Dashboard:	6
Forum:	6
3. VORGEHENSWEISE	8
Log-in und Register Formular:	8
Register:	Error! Bookmark not defined.
Login:	Error! Bookmark not defined.
Main Page:	23
Course Page:	26
User Dashboard:	30
Forum:	36
Discussion.php:	36
reply.php:	44
comment.php:	46
Datenbank:	48
4. VERWENDETE BIBLIOTHEKEN	49
5. DESIGN	49
Allgemein	49

Nutzen.....	49
Wo wurde es implementiert?	49
Nutzen von CSS	49
Wieso ist Design wichtig?	49

1. Planung

Filipe:

Register, Forum, User Content:

Registrierung Seite, wo folgendes nötig ist:

- Benutzer Name
- Email
- Passwort
- Passwort Wiederholung

Forum Seite, wo folgendes nötig ist:

- Benutzer Name Rückgabe
- Kommentar erstellen
- Darauf antworten können

Deni:

Log-in/Log-out, User Dashboard, CSS/Styling:

Login Seite, wo folgendes nötig ist:

- Existierende E-Mail
- Das Passwort der E-Mail

User Dashboard, wo folgendes angezeigt wird:

- Benutzer Name
- E-Mail
- Datum der Erstellung des Accounts

CSS:

- Generelle Kompatibilität mit mobilen Geräten
- Erweiterung des Design templates (resize Möglichkeit, usw)
- Log-in und Register Design
- User Dashboard 50%

Lou:

Mainpage, CSS/Styling, Session, Designtemplate, topnav

Mainpage:

- Design
- Setup
- Content

CSS:

- Mainpage (except desktop to mobile resizing)
- Forum
- Cours
- Dashboard 50%

Session:

- (If logged in, only log out button and no signin/login)
=> Every page

Alexandro:

Course1page, Course2page, Webpages-Examples

Course1page:

- Content
- Verteilung der kapiteln in Blocks
- Text mit links
- Interaktive Knöpfe

Course2page:

- Content
- Verteilung des Cours in Blocks
- Examples Webpages
- Text mit links
- Interaktive Knöpfe

2.Struktur

Log-in und Register Formular:

Register:

Code Methoden.....

Login:

Das Login Formular besteht aus einem Fenster und zwei separaten Buttons. Im Fenster sind zwei Text Boxen, welche den User nach einer existierenden E-Mail und dem dazugehörigen Password fragt. Darunter ein SUBMIT Button um die Daten dem Server zu schicken, damit man sich einloggen kann.

Die zwei separaten Buttons bestehen aus Redirects. Einmal zur Home page und zum Register

Main Page:

Top Navigation:

Wenn der \$_SESSION[“user”] Parameter gesetzt ist soll nur die “Log out” Option während, wenn der Parameter nicht gesetzt wurde, die Optionen “Sign in” und “Log in” angezeigt werden sollen.

Information Cards:

Alle Informationen werden in Divs wiedergegeben die Kartei Karten ähneln sollen. Diese werden auf der ganzen Seite verwendet.

Contributors:

Hier stehen alle Beteiligten der Webseite und ihren Arbeitsbereichen am Projekt.

Course Page:

Top Navigation:

Wenn der `$_SESSION["user"]` Parameter gesetzt ist soll nur die "Log out" Option während, wenn der Parameter nicht gesetzt wurde, die Optionen "Sign in" und "Log in" angezeigt werden sollen.

Set-up:

Content verteilt in verschiedene Blöcke pro Kapitel auf den 2 Webseiten-Cours. An der Unterseite, ein Block mit "previous" and "next" Knöpfen um von der eine Webseite-Cours zu der andere gehen.

Display:

Links und interaktive Knöpfe für den User zu navigieren und Beispiele-code und -webseiten zum Sehen und Benutzen. Text zentriert und gut verteilt in den Blöcke.

User Dashboard:

Top Navigation:

Wenn der `$_SESSION["user"]` Parameter gesetzt ist soll nur die "Log out" Option während, wenn der Parameter nicht gesetzt wurde, die Optionen "Sign in" und "Log in" angezeigt werden sollen.

Profile Card:

Ein Standard Profil Bild wird angezeigt mit dem vollen Namen des Users

Navigation Card:

Die Navigation Card zeigt an in welcher Card man sich befindet

Information Card:

Die Information Card zeigt an welchen vollen Namen der User besitzt, dazu noch die E-Mail Adresse und zum Schluss das Erstellungsdatum des Users

Forum:


Top Navigation:

Wenn der \$_SESSION["user"] Parameter gesetzt ist soll nur die "Log out" Option während, wenn der Parameter nicht gesetzt wurde, die Optionen "Sign in" und "Log in" angezeigt werden sollen.

3. Vorgehensweise

Log-in und Register Formular:

Register:

 register.php

```
<?php //Deni Code
session_start();
if(isset($_SESSION["user"]))
{
    header("Location: userDashboard.php");
}
?>
```

Dieses PHP-Skript dient zur Verwaltung von Benutzersitzungen und zur Weiterleitung authentifizierter Benutzer auf das Dashboard. Dies ist ein gängiges Muster in Webanwendungen, um sicherzustellen, dass bereits angemeldete Benutzer

direkt zu ihrer Hauptseite geleitet werden, ohne sich erneut anmelden zu müssen.

- **Funktionsaufruf:** `session_start()`
 - **Zweck:** Startet eine neue Sitzung oder setzt eine bestehende Sitzung fort (eine Sitzung ist eine Möglichkeit, Daten über mehrere Seitenaufrufe hinweg zu speichern).
 - **Mechanismus:** Überprüft, ob eine Sitzungs-ID (eine eindeutige Kennung für die Sitzung) über eine GET/POST-Anfrage (Daten, die vom Browser an den Server gesendet werden) oder ein Cookie (kleine Datei, die Daten im Browser speichert) übermittelt wird. Wenn keine Sitzungs-ID gefunden wird, erstellt PHP eine neue.
 - **Sitzungs-ID:** Eine eindeutige Kennung, die verwendet wird, um die Sitzungsdaten auf dem Server abzurufen.
-

- **Benutzerzustandsverwaltung:** Durch das Starten einer Sitzung kann die Anwendung benutzerspezifische Daten (wie Benutzername oder Benutzer-ID) über mehrere Seiten hinweg speichern und beibehalten.
 - **Sitzungsinitialisierung:** Stellt sicher, dass sitzungsbezogene Daten im gesamten Skript verfügbar sind, sodass nachfolgende Sitzungsüberprüfungen und -operationen durchgeführt werden können.
 - **isset()-Funktion:** `isset($_SESSION["user"])`
 - **Zweck:** Überprüft, ob die Variable `$_SESSION["user"]` gesetzt ist und nicht `null` ist (stellt sicher, dass die Variable existiert und einen Wert hat).
 - **Superglobale Variable:** `$_SESSION` ist ein superglobales Array (eine spezielle Art von Variable, die überall im Skript verfügbar ist), das Sitzungsvariablen speichert.
 - **Benutzerauthentifizierung:** Diese Überprüfung stellt fest, ob der Benutzer angemeldet ist, indem überprüft wird, ob die Variable `$_SESSION["user"]` existiert.
 - **Zugriffskontrolle:** Stellt sicher, dass nur authentifizierte Benutzer (Benutzer, die sich angemeldet haben) auf bestimmte Seiten oder Funktionen zugreifen können. Wenn die Benutzersitzung vorhanden ist, bedeutet dies, dass der Benutzer angemeldet ist.
-
- **header()-Funktion:** `header("Location: userDashboard.php")`
 - **Zweck:** Sendet einen HTTP-Header (eine Information, die an den Browser gesendet wird) an den Client (Browser), der ihn anweist, zu `userDashboard.php` zu navigieren.
 - **HTTP-Statuscode:** Standardmäßig wird ein `302 Found`-Statuscode (eine Nachricht, die den Browser informiert, dass die Seite vorübergehend verschoben wurde) gesendet, der eine temporäre Weiterleitung anzeigt.
 - Leitet angemeldete Benutzer automatisch zu ihrem Dashboard weiter, um zu vermeiden, dass sie nach dem Anmelden manuell navigieren müssen.
 - Durch die Verwendung von `header("Location: userDashboard.php")` wird sichergestellt, dass der Benutzer sofort zur richtigen Seite weitergeleitet wird.
-

Dieses HTML-Dokument enthält die Struktur und den Stil eines Registrierungsformulars. Es umfasst Verweise auf externe CSS-Dateien für das Design und enthält JavaScript-Funktionen für die Navigation und den Vollbildmodus.

- **Meta-Tags:**

- `<meta charset="UTF-8">`: Stellt die Zeichenkodierung

auf UTF-8 ein, was die Darstellung der meisten Zeichen in den meisten Sprachen ermöglicht.

- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Setzt die Breite des Ansichtsfensters auf die Breite des Geräts und stellt sicher, dass die Seite auf mobilen Geräten richtig skaliert wird.

- **Titel:**

- `<title>Registration Form</title>`: Gibt den Titel der Webseite an, der in der Registerkarte des Browsers angezeigt wird.

- **Link-Tags:**

- `<link rel="icon" href="../Images/icons/network_internet_pcs_installer-1.png" type="image/x-icon">`: Setzt das Favicon (kleines Icon, das in der Browser-Registerkarte angezeigt wird).
- `<link rel="stylesheet" href="https://unpkg.com/98.css">`: Verlinkt eine externe CSS-Datei für das Design der Seite.
- `<link rel="stylesheet" href="../CSS/RegisterAndLoginStyles.css">`: Verlinkt eine weitere CSS-Datei, die spezifische Stile für das Registrierungs- und Login-Formular enthält.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration Form</title>
  <link rel="icon" href="../Images/icons/network_internet_pcs_installer-1.png" type="image/x-icon">
  <link rel="stylesheet" href="https://unpkg.com/98.css">
  <link rel="stylesheet" href="../CSS/RegisterAndLoginStyles.css"> <!-- Lou Code -->

  <!-- Deni Code -->
  <script>
    function redirectToHome() {
      console.log("Redirecting to home...");
      window.location.href = "../index.php";
    }

    function toggleFullscreen() {
      if (!document.fullscreenElement) { //if document isn't in fullscreen
        document.documentElement.requestFullscreen().catch(err => {
          //give error with the error message attached
          alert("Error attempting to enable full-screen mode: ${err.message} (${err.name})");
        });
      } else {
        if (document.exitFullscreen) {
          //document
          document.exitFullscreen();
        }
      }
    }
  </script>
</head>
```

Funktionen:

1. `redirectToHome()`

- **Zweck:** Leitet den Benutzer zur Startseite (`index.php`) weiter.
 - **Funktionsweise:**
 - `console.log("Redirecting to home...");` : Gibt eine Nachricht in der Browser-Konsole aus, um anzuzeigen, dass die Weiterleitung erfolgt.
 - `window.location.href="../index.php";` : Ändert die URL der aktuellen Seite zur Startseite.
-

2. `toggleFullscreen()`

- **Zweck:** Schaltet den Vollbildmodus ein und aus.
 - **Funktionsweise:**
 - `if (!document.fullscreenElement):` Überprüft, ob das Dokument sich nicht im Vollbildmodus befindet.
 - `document.documentElement.requestFullscreen().catch(err => { ... }):` Versucht, den Vollbildmodus zu aktivieren und gibt im Fehlerfall eine Warnung aus.
 - `else:` Wenn das Dokument bereits im Vollbildmodus ist:
 - `if (document.exitFullscreen):` Überprüft, ob die Methode `exitFullscreen` verfügbar ist.
 - `document.exitFullscreen();` : Beendet den Vollbildmodus.
-

```

<body>
<div class="container">
  <?php
  //checks if everything is set
  if (isset($_POST["submit"])) {
    $fullName = $_POST["fullname"];
    $email = $_POST["email"];
    $password = $_POST["password"];
    $passwordRepeat = $_POST["repeat_password"];

    $password_hash = password_hash($password, PASSWORD_DEFAULT);

    $errors = array();

    //adds element to the error array
    if(empty($fullName) OR empty($email) OR empty($password) OR empty($passwordRepeat)){
      array_push($errors,"All fields are required");
    }
    if(!filter_var($email,FILTER_VALIDATE_EMAIL)){
      array_push($errors,"Email is not valid");
    }
    if(strlen($password)< 8){
      array_push($errors,"Password must be 8 characters long");
    }
    if($password !== $passwordRepeat){
      array_push($errors,"Password does not match!");
    }

    require_once "database.php";
  }

```

Dieser Abschnitt des HTML-Dokuments enthält ein PHP-Skript, das die Benutzereingaben bei der Registrierung überprüft und validiert. Es prüft, ob alle erforderlichen Felder ausgefüllt sind, ob die E-Mail-Adresse gültig ist, ob das Passwort lang genug ist und ob die Passwörter übereinstimmen.

```
if (isset($_POST["submit"]))
```

- **Zweck:** Überprüft, ob das Formular durch Drücken des "Submit"-Buttons abgeschickt wurde.
- **Mechanismus:** `isset()` prüft, ob die Variable `$_POST["submit"]` gesetzt ist, was bedeutet, dass das Formular abgeschickt wurde.

```

$fullName = $_POST["fullname"];
$email = $_POST["email"];
$password = $_POST["password"];
$passwordRepeat = $_POST["repeat_password"];

```

- **Zweck:** Speichert die vom Benutzer eingegebenen Daten in Variablen.
- **Superglobale Variable:** `$_POST` ist ein superglobales Array, das die Daten des abgeschickten Formulars enthält.

```
$password_hash = password_hash($password, PASSWORD_DEFAULT);
```

- **Zweck:** Verschlüsselt das Passwort, bevor es in der Datenbank gespeichert wird.
- **Funktion:** `password_hash()` generiert einen sicheren Hash des Passworts mit dem Standardalgorithmus (`PASSWORD_DEFAULT`).

```
$errors = array();
```

- **Zweck:** Erstellt ein leeres Array, um Fehlermeldungen zu speichern.

```
if(empty($fullName) OR empty($email) OR empty($password) OR empty($passwordRepeat)){
    array_push($errors, "All fields are required");
}
```

- **Zweck:** Überprüft, ob eines der Felder leer ist.
- **Mechanismus:** `empty()` prüft, ob eine Variable leer ist. Wenn ja, wird eine Fehlermeldung zum Fehler-Array hinzugefügt.

```
if(!filter_var($email, FILTER_VALIDATE_EMAIL)){
    array_push($errors, "Email is not valid");
}
```

- **Zweck:** Überprüft, ob die eingegebene E-Mail-Adresse gültig ist.
- **Mechanismus:** `filter_var()` prüft die E-Mail-Adresse mit dem Filter `FILTER_VALIDATE_EMAIL`. Wenn die E-Mail ungültig ist, wird eine Fehlermeldung hinzugefügt.

```
if(strlen($password) < 8){
    array_push($errors, "Password must be 8 characters long");
}
```

- **Zweck:** Überprüft, ob das Passwort mindestens 8 Zeichen lang ist.
- **Mechanismus:** `strlen()` gibt die Länge der Zeichenkette zurück. Wenn das Passwort kürzer als 8 Zeichen ist, wird eine Fehlermeldung hinzugefügt.

```
if($password !== $passwordRepeat){
    array_push($errors, "Passwords do not match!");
}
```

- **Zweck:** Überprüft, ob die beiden eingegebenen Passwörter übereinstimmen.
- **Mechanismus:** Ein Vergleichsoperator (`!==`) prüft, ob die Passwörter unterschiedlich sind. Wenn ja, wird eine Fehlermeldung hinzugefügt.

```
require_once "database.php";
```

- **Zweck:** Stellt sicher, dass das Skript die Datei `database.php` lädt, um die Verbindung zur Datenbank herzustellen.

- **Mechanismus:** `require_once` stellt sicher, dass die Datei nur einmal eingebunden wird, selbst wenn sie mehrfach referenziert wird.

```
//if errors array is empty, no informations are being put in the database
if(count($errors)>0){
    foreach($errors as $error){
        echo "<div class='alert alert-danger'>$error</div>";
    }
}else{

    //? = placeholder
    $sql = "INSERT INTO users (dtName, dtEmail, dtPassword) VALUES ( ?, ?, ?)";
    $stmt = mysqli_stmt_init($connection);
    //gives true or false
    $prepareStmt = mysqli_stmt_prepare($stmt,$sql);
    if($prepareStmt){
        mysqli_stmt_bind_param($stmt,"sss",$fullName,$email,$password_hash);
        mysqli_stmt_execute($stmt);
        echo "<div class='alert alert-success'>You are registered successfully.</div>";
    }else{
        die("Something went wrong");
    }
}
?>
```

Dieser Abschnitt des PHP-Skripts validiert die Benutzereingaben und fügt die Daten in die Datenbank ein, wenn keine Fehler vorliegen. Es zeigt auch entsprechende Fehlermeldungen oder Erfolgsmeldungen an.

```
if(count($errors) > 0){
    foreach($errors as $error){
        echo "<div class='alert alert-danger'>$error</div>";
    }
} else {
```

- **Zweck:** Überprüft, ob das Fehler-Array Einträge enthält. Wenn ja, werden die Fehlermeldungen angezeigt.
- **Mechanismus:**
 - `count($errors) > 0`: Überprüft, ob das Fehler-Array mehr als 0 Einträge enthält.
 - `foreach($errors as $error)`: Iteriert durch das Fehler-Array und zeigt jede Fehlermeldung innerhalb eines HTML-DIVs an.

```
$sql = "INSERT INTO users (dtName, dtEmail, dtPassword) VALUES (?, ?, ?)";
$stmt = mysqli_stmt_init($connection);

// Bereitet die SQL-Anweisung vor
$prepareStmt = mysqli_stmt_prepare($stmt, $sql);
```

- **Zweck:** Bereitet die SQL-Anweisung für das Einfügen der Benutzerdaten in die Datenbank vor.
- **Mechanismus:**
 - `$sql`: Enthält die SQL-Abfrage mit Platzhaltern (?), die durch die tatsächlichen Werte ersetzt werden.
 - `mysqli_stmt_init($connection)`: Initialisiert eine neue MySQLi-Statement-Instanz.

- o `mysqli_stmt_prepare($stmt, $sql)`: Bereitet die SQL-Anweisung vor und überprüft, ob sie korrekt ist.

```
if($prepareStmt){

    mysqli_stmt_bind_param($stmt, "sss", $fullName, $email, $password_hash);

    mysqli_stmt_execute($stmt);

    echo "<div class='alert alert-success'>You are registered successfully.</div>";

} else {

    die("Something went wrong");

}
```

- **Zweck:** Bindet die tatsächlichen Werte an die Platzhalter in der SQL-Anweisung und führt die Anweisung aus.
 - **Mechanismus:**
 - o `mysqli_stmt_bind_param($stmt, "sss", $fullName, $email, $password_hash)`: Bindet die Variablen `$fullName`, `$email` und `$password_hash` an die vorbereitete Anweisung. Der Typ der gebundenen Parameter wird durch "sss" angegeben (drei Strings).
 - o `mysqli_stmt_execute($stmt)`: Führt die vorbereitete und gebundene SQL-Anweisung aus.
 - o `echo "<div class='alert alert-success'>You are registered successfully.</div>";`: Zeigt eine Erfolgsmeldung an, wenn die Anweisung erfolgreich ausgeführt wurde.
 - o `else { die("Something went wrong"); }`: Gibt eine Fehlermeldung aus und beendet das Skript, falls die Anweisung nicht erfolgreich vorbereitet werden konnte.
-


```

<form action="register.php" method="post"> <!--Deni & Filip Code-->
  <div class="window" style="height: 480px;">
    <div class="title-bar">
      <div class="title-bar-img">
        
      </div>
      <div class="title-bar-text">Register</div>
      <div class="title-bar-controls">
        <!--Deni Code-->
        <button aria-label="Minimize" type="button" onclick="redirectToHome()"></button>
        <button aria-label="Maximize" type="button" onclick="toggleFullscreen()"></button>
        <button aria-label="Close" type="button" onclick="redirectToHome()"></button>
      </div>
    </div>
    <div class="window-body">
      <p style="text-align: left; margin-top: 1px; font-size: 18px; margin-bottom: 5px">Full Name:</p>
      <div class="form-group">
        <input type="text" class="form-control" name="fullname" placeholder="Full Name:" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <p style="text-align: left; font-size: 18px; margin-bottom: 5px">Email:</p>
      <div class="form-group">
        <input type="email" class="form-control" name="email" placeholder="Email:" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <p style="text-align: left; font-size: 18px; margin-bottom: 5px">Password:</p>
      <div class="form-group">
        <input type="password" class="form-control" name="password" placeholder="Password:" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <p style="text-align: left; font-size: 18px; margin-bottom: 5px">Repeat Password:</p>
      <div class="form-group">
        <input type="password" class="form-control" name="repeat_password" placeholder="Repeat Password:" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <div class="form-btn">
        <input type="submit" class="btn btn-primary" value="Register" name="submit" style="padding: 18px; font-size: 18px; margin-top: 20px;">
      </div>
    </div>
  </form>
  <!--deni--><div style="font-size: 15px;"><p>Already registered? <a href="login.php">Log-in here!</a></p></div> <!--Deni Code-->
</div>
</body>
</html>

```

Dieser HTML-Code enthält das Formular für die Registrierung von Benutzern. Er besteht aus mehreren Eingabefeldern zur Eingabe des vollständigen Namens, der E-Mail-Adresse und des Passworts sowie Funktionen zum Einreichen der Daten und zur Navigation.

• Formularstruktur:

- `<form action="register.php" method="post">`: Definiert das Formular, das per POST-Methode an "register.php" gesendet wird.
- `<div class="window" style="height: 480px;">`: Definiert das Hauptfenster des Formulars mit einer festen Höhe.

• Titelzeile:

- `<div class="title-bar">`: Definiert die Titelleiste des Fensters.
- `<div class="title-bar-img">`: Beinhaltet das Bildsymbol.
- `<div class="title-bar-text">Register</div>`: Beinhaltet den Titeltext.
- `<div class="title-bar-controls">`: Beinhaltet die Steuerungsknöpfe für Minimieren, Maximieren und Schließen.

- **Formulareingabefelder:**

- `<input type="text" class="form-control" name="fullname" placeholder="Full Name: ": Eingabefeld für den vollständigen Namen.`
 - `<input type="email" class="form-control" name="email" placeholder="Email: ": Eingabefeld für die E-Mail-Adresse.`
 - `<input type="password" class="form-control" name="password" placeholder="Password: ": Eingabefeld für das Passwort.`
 - `<input type="password" class="form-control" name="repeat_password" placeholder="Repeat Password: ": Eingabefeld für die Wiederholung des Passworts.`
-

- **Einreichen des Formulars:**

- `<input type="submit" class="btn btn-primary" value="Register" name="submit" style="padding: 18px; font-size: 18px; margin-top: 20px; ">: Einreichen-Knopf für das Formular.`
-

- **Registrierungslink:**

- `<div style="font-size: 15px; "><p>Already registered? Log-in here!</p></div>: Link zur Login-Seite für bereits registrierte Benutzer.`
-

Login :

login.php

```
<?php //Deni Code
session_start();
if(isset($_SESSION["user"]))
{
    header("Location: userDashboard.php");
}
?>
```

Dieselbe PHP-Sektion wurde auch im login.php eingefügt, sodass der User sich nicht nochmal unnötig anmelden muss. Stattdessen wird er auch zur userDashboard.php Seite durch die header(); Funktion geschickt nachdem eine Session gestatet

wird und falls die Variable \$_SESSION["user"] gesetzt ist und nicht null ist (stellt sicher, dass die Variable existiert und einen Wert hat).

Der folgende Code Snippet enthält den html <head> des login.php files:

```
11 <!--Deni Code-->
12 <!DOCTYPE html>
13 <html lang="en">
14     <head>
15         <meta charset="UTF-8">
16         <meta name="viewport" content="width=device-width, initial-scale=1.0">
17         <title>Login</title>
18         <link rel="icon" href="../Images/icons/users_key-5.png" type="image/x-icon">
19         <link rel="stylesheet" href="https://unpkg.com/98.css">
20         <link rel="stylesheet" href="../CSS/RegisterAndLoginStyles.css">
21         <script src="../JS/registerButton.js"></script>
22     </head>
```

- **Titel:**

- `<title>Login</title>`: Gibt den Titel der Webseite an, der in der Registerkarte des Browsers angezeigt wird.

- **Link-Tags:**

```
<link rel="icon" href="../Images/icons/users_key-5.png" type="image/x-icon">
```

- Diese Zeile verlinkt ein Favicon für die Webseite. Das Favicon ist das kleine Symbol, das im Tab des Browsers neben dem Seitentitel angezeigt wird.
- `href="../Images/icons/users_key-5.png"` gibt den Pfad zur Bilddatei des Favicons an.
- `type="image/x-icon"` gibt den Dateityp des Favicons an.

```
<link rel="stylesheet" href="https://unpkg.com/98.css">
```

- Diese Zeile verlinkt eine externe CSS-Datei.
- `href="https://unpkg.com/98.css"` gibt die URL der CSS-Datei an, die das Aussehen und Layout der Webseite beeinflusst.

```
<link rel="stylesheet" href="../CSS/RegisterAndLoginStyles.css">
```

- Diese Zeile verlinkt eine lokale CSS-Datei, die sich im Verzeichnis ../CSS/ befindet.
- href="../CSS/RegisterAndLoginStyles.css" gibt den Pfad zu dieser CSS-Datei an, die spezifische Stile für die Registrierungs- und Anmeldeseiten enthält.

```
<script src="../JS/registerButton.js"></script>
```

- Diese Zeile verlinkt eine lokale JavaScript-Datei, die sich im Verzeichnis ../JS/ befindet.
- src="../JS/registerButton.js" gibt den Pfad zu dieser JavaScript-Datei an, die wahrscheinlich Funktionen für den Registrierungsbutton enthält.

Der folgende Code Snippet enthält die PHP Sektion des html <body> des login.php:

```

23 <body>
24 <div class="container">
25 <?php
26     if(isset($_POST["login"]))
27     {
28         $email = $_POST["email"];
29         $password = $_POST["password"];
30
31         require_once "database.php";
32         $sql = "SELECT * FROM users WHERE dtEmail = '$email'";
33         $result = mysqli_query($connection, $sql);
34         $user = mysqli_fetch_array($result, MYSQLI_ASSOC);
35         if($user) {
36             if(password_verify($password, $user["dtPassword"])) {
37                 session_start();
38                 $_SESSION["user"] = $user["dtName"];
39                 header("Location: userDashboard.php");
40                 die();
41             }else{
42                 echo "<div class='alert alert-danger'>Email or Password does not match</div>";
43             }
44         }else {
45             echo "<div class='alert alert-danger'>Email or Password does not match</div>";
46         }
47     }
48 >?>
49

```

• Prüfen auf Login-Submission:

```
if (isset($_POST["login"])) {
```

- Diese Zeile überprüft, ob das Login-Formular abgesendet wurde.

• Variablenzuweisung:

```
$email = $_POST["email"];
$password = $_POST["password"];
```

- Die eingegebenen E-Mail- und Passwortwerte werden aus dem Formular abgerufen und in den Variablen \$email und \$password gespeichert.

• Datenbankverbindung:

```
require_once "database.php";
```

- Hier wird die Datei database.php eingebunden, welche wahrscheinlich die Verbindung zur Datenbank herstellt.

- **SQL-Abfrage:**

```
$sql = "SELECT * FROM users WHERE dtEmail = '$email'";
$result = mysqli_query($connection, $sql);
$user = mysqli_fetch_array($result, MYSQLI_ASSOC);
```

- Eine SQL-Abfrage wird ausgeführt, um den Benutzer mit der angegebenen E-Mail-Adresse in der Datenbank zu finden.
- Das Ergebnis der Abfrage wird in `$result` gespeichert und dann in ein assoziatives Array `$user` umgewandelt.

- **Überprüfen des Benutzers:**

```
if ($user) {
```

- Es wird überprüft, ob ein Benutzer mit der angegebenen E-Mail-Adresse gefunden wurde.

- **Passwortverifizierung:**

```
if (password_verify($password, $user['dtPassword'])) {
```

- Das eingegebene Passwort wird mit dem in der Datenbank gespeicherten Passwort (welches vermutlich gehasht ist) verglichen.

- **Sitzungsstart und Weiterleitung:**

```
session_start();
$_SESSION["user"] = $user["dtName"];
header("Location: userDashboard.php");
die();
```

- Wenn das Passwort korrekt ist, wird eine Sitzung gestartet und der Benutzername in der Sitzung gespeichert.
- Anschließend wird der Benutzer zur `userDashboard.php` Seite weitergeleitet und das Skript wird beendet.

- **Fehlermeldungen:**

```
echo "<div class='alert alert-danger'>Email or Password does not match</div>";
```

- Falls die E-Mail-Adresse nicht gefunden wird oder das Passwort nicht übereinstimmt, wird eine Fehlermeldung angezeigt.
-

Der folgende Code Snippet enthält den Rest des html <body> des login.php:

```
<form action="login.php" method="post">
  <div class="window">
    <div class="title-bar">
      <div class="title-bar-img">
        
      </div>
      <div class="title-bar-text">Login</div>
      <div class="title-bar-controls">
        <!-- Deni Code -->
        <button aria-label="Minimize" type="button" onclick="redirectToHome()"></button>
        <button aria-label="Maximize" type="button" onclick="toggleFullscreen()"></button>
        <button aria-label="Close" type="button" onclick="redirectToHome()"></button>
      </div>
    </div>
    <div class="window-body">
      <p style="text-align: left; margin-top: 1px; font-size: 18px; margin-bottom: 5px;">Email:</p>
      <div class="form-group">
        <input type="email" placeholder="Enter Email:" name="email" class="form-control" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <p style="text-align: left; font-size: 18px; margin-bottom: 5px;">Password:</p>
      <div class="form-group">
        <input type="password" placeholder="Enter Password:" name="password" class="form-control" style="padding: 15px; font-size: 15px; margin-top: 10px;">
      </div>

      <div class="form-btn">
        <input type="submit" value="Login" name="login" class="btn btn-primary" style="padding: 18px; font-size: 18px; margin-top: 20px;">
      </div>
    </div>
  </div>
</form>
<div class="form-btn" style="font-size: 15px;">
  <button onclick="location.href = 'register.php';" class="btn btn-primary">Register</button>
  <br>
  <button onclick="location.href = '../index.php';" class="btn btn-primary">Back</button>
</div>
</div>
</body>
</html>
```

```
<form action="login.php" method="post">
```

- Dieses Formular sendet die eingegebenen Daten mittels POST-Methode an die login.php-Datei.

• Hauptteil des Formulars

<div> mit der Klasse window:

• Titelzeile (title-bar):

- Enthält das Bild und den Text für die Login-Seite.
- title-bar-controls enthält drei Buttons (Minimieren, Maximieren und Schließen), die JavaScript-Funktionen aufrufen.

• Hauptkörper (window-body):

- Email-Eingabefeld:

```
<p style="text-align: left; margin-top: 1px; font-size: 18px; margin-bottom: 5px;">Email:</p>
<div class="form-group">
  <input type="email" placeholder="Enter Email:" name="email"
  class="form-control" style="padding: 15px; font-size: 15px; margin-top: 10px;">
</div>
```

Label und Eingabefeld für die E-Mail-Adresse des Benutzers.

- Passwort-Eingabefeld:

```
<p style="text-align: left; font-size: 18px; margin-bottom: 5px">Password:</p>
<div class="form-group">
  <input type="password" placeholder="Enter Password:" name="password"
class="form-control" style="padding: 15px; font-size: 15px; margin-top: 10px;">
</div>
```

Label und Eingabefeld für das Passwort des Benutzers.

- Login-Button:

```
<div class="form-btn">
  <input type="submit" value="Login" name="login" class="btn btn-
primary" style="padding: 18px; font-size: 18px; margin-top: 20px;">
</div>
```

Ein submit-Button, der das Formular absendet.

Unterhalb des Formulars gibt es weitere Buttons für die Navigation.

```
<div class="form-btn">
  <button onclick="location.href = 'register.php';" class="btn btn-
primary">Register</button>
  <button onclick="location.href = '../index.php';" class="btn btn-
primary">Back</button>
</div>
```

- **Register-Button:** Leitet zur Registrierungsseite weiter.
- **Back-Button:** Leitet zur Startseite oder vorherigen Seite zurück.

Main Page :

1. Session starten

Zweck:

- Eine Sitzung wird gestartet, um Benutzerdaten zwischen verschiedenen Seitenaufrufen zu speichern und abzurufen.

Mechanismus:

- `session_start()` initialisiert eine neue oder setzt eine bestehende Sitzung fort. Ohne diese Zeile könnten keine Session-Variablen gesetzt oder ausgelesen werden.
-

2. Überprüfung, ob der Benutzer eingeloggt ist

Zweck:

- Es wird überprüft, ob der Benutzer bereits eingeloggt ist, um entsprechend die richtige Navigationsleiste anzuzeigen.

Mechanismus:

- `isset($_SESSION["user"])` prüft, ob die Session-Variable `user` gesetzt ist. Wenn diese Variable nicht existiert, bedeutet das, dass der Benutzer nicht eingeloggt ist.
-

3. Navigationsleiste für nicht eingeloggte Benutzer

Zweck:

- Eine Navigationsleiste mit Optionen für nicht eingeloggte Benutzer anzeigen.

Mechanismus:

- Es wird HTML-Code generiert, um die Navigationsleiste mit Links zu verschiedenen Seiten anzuzeigen. Die Links "Sign up" und "Log in" sind für nicht eingeloggte Benutzer sichtbar.
-

4. Navigationsleiste für eingeloggte Benutzer

Zweck:

- Eine Navigationsleiste mit Optionen für eingeloggte Benutzer anzeigen.

Mechanismus:

- Es wird HTML-Code generiert, um die Navigationsleiste mit Links zu verschiedenen Seiten anzuzeigen. Der "Home"-Link wird hervorgehoben und der "Log-out"-Link ist für eingeloggte Benutzer sichtbar.

```
<?php
session_start();
if(!isset($_SESSION["user"]))
{
    echo '<p style="margin: 0px; padding-top: 10px; background-color: #1A4D2E; text-align: center; color: white;">Welcome to our website</p>';
    echo '<div class="topnav">';
    echo '<div class="row">';
    echo '<div class="navleft">';
    echo '<a class="active" href="index.php">Home</a>';
    echo '<a href="Includes/templatedot.php">Course</a>';
    echo '<a href="Includes/userDashboard.php">Dashboard</a>';
    echo '<a href="Includes/discussion.php">Forum</a>';
    echo '</div>';
    echo '<div class="navright">';
    echo '<a href="Includes/register.php">Sign up</a>';
    echo '<a href="Includes/login.php">Log in</a>';
    echo '</div>';
    echo '</div>';
    echo '</div>';
}
else {
    echo '<p style="margin: 0px; padding-top: 10px; background-color: #1A4D2E; text-align: center; color: white;">Welcome to our website</p>';
    echo '<div class="topnav">';
    echo '<div class="row">';
    echo '<div class="navleft">';
    echo '<a class="active" href="index.php" style="background-color: #E8DFCA; color: black;">Home</a>';
    echo '<a href="Includes/templatedot.php">Course</a>';
    echo '<a href="Includes/userDashboard.php">Dashboard</a>';
    echo '<a href="Includes/discussion.php">Forum</a>';
    echo '</div>';
    echo '<div class="navright">';
    echo '<a href="Includes/logout.php">Log-out</a>';
    echo '</div>';
    echo '</div>';
    echo '</div>';
}
>>
```

5. JavaScript für fade Animation

```
1 //Deni
2
3 const observer = new IntersectionObserver((entries) => {
4     entries.forEach((entry) => {
5         console.log(entry);
6         if(entry.isIntersecting) {
7             entry.target.classList.add('rowShow');
8         }
9         else {
10            entry.target.classList.remove('rowShow');
11        }
12    });
13 });
14
15 const hiddenElements = document.querySelectorAll('.row');
16 hiddenElements.forEach(el => observer.observe(el));
```

- **Erstellung eines IntersectionObservers:**

- `IntersectionObserver`: Ein Konstruktor, der eine neue Instanz des `IntersectionObserver`-Objekts erstellt. Dieses Objekt überwacht die Sichtbarkeit von Elementen im Viewport.
- `(entries) => { ... }`: Eine Callback-Funktion, die aufgerufen wird, wenn sich die Sichtbarkeit der beobachteten Elemente ändert. Die `entries`-Parameter enthält eine Liste von `IntersectionObserverEntry`-Objekten, die Informationen über die Sichtbarkeit der beobachteten Elemente enthalten.
- `entries.forEach((entry) => { ... })`: Für jede Änderung der Sichtbarkeit wird eine Funktion aufgerufen, die:
 - `console.log(entry)`: Das `entry`-Objekt in der Konsole ausgibt, das Informationen über das beobachtete Element enthält.
 - `if(entry.isIntersecting) { ... }`: Überprüft, ob das Element sichtbar ist (`isIntersecting` ist `true`), und wenn ja:
 - `entry.target.classList.add('rowShow')`: Fügt die Klasse `rowShow` zum sichtbaren Element hinzu.
 - `else { ... }`: Wenn das Element nicht sichtbar ist:
 - `entry.target.classList.remove('rowShow')`: Entfernt die Klasse `rowShow` vom nicht sichtbaren Element.

- **Auswahl der zu beobachtenden Elemente:**

```
const hiddenElements = document.querySelectorAll('.row');
```

- `document.querySelectorAll('.row')`: Wählt alle Elemente im Dokument aus, die die Klasse `row` haben, und speichert sie in der Variable `hiddenElements`.

- **Start der Beobachtung:**

```
hiddenElements.forEach((el) => observer.observe(el));
```

- `hiddenElements.forEach((el) => { ... })`: Für jedes ausgewählte Element:
 - `observer.observe(el)`: Beginnt der `IntersectionObserver`, das Element zu beobachten, um Änderungen seiner Sichtbarkeit zu verfolgen.
-

Course Page:

Planung:

Recherchen zum Cours-Inhalt, mögliche Beispiele und Aufgaben für den Benutzer. Planung der Struktur und Aufteilung der Kapiteln.

Nutzen von benutzen CSS mit Sitzung-Änderung(log in):

```
<?php
session_start();
if(!isset($_SESSION["user"]))
{
    echo '<input onclick="toggleNav()" type="checkbox" id="nav_check" hidden>';
    echo '<label for="nav_check" class="hamburger" id="hamburger">';
    echo '<span>M</span><span>E</span><span>N</span><span>U</span>';
    echo '</label>';
    echo '<p class="welcome">Welcome to our website</p>';
    echo '<div id="mySidenav" class="topnav">';
    echo '<div class="navrow">';
    echo '<div class="navleft">';
    echo '<h2>HowToWebsite</h2>';
    echo '<a href=".."index.php">Home</a>';
    echo '<a class="active" href="templatedot.php">Course</a>';
    echo '<a href="userDashboard.php">Dashboard</a>';
    echo '<a href="discussion.php">Forum</a>';
    echo '<a href="register.php">Sign up</a>';
    echo '<a href="login.php">Log in</a>';
    echo '</div>';
    echo '</div>';
    echo '</div>';
}
```

Oberer Teil der Webseite bleibt im Modus nicht eingeloggt, also mit sign-up und log-in Knöpfe. Plus die generelle Structur --> Homepage, Cours,...und etc. Knöpfe + Titelen.

```
}else {
    echo '<input onclick="toggleNav()" type="checkbox" id="nav_check" hidden>';
    echo '<label for="nav_check" class="hamburger" id="hamburger">';
    echo '<span>M</span><span>E</span><span>N</span><span>U</span>';
    echo '</label>';
    echo '<p class="welcome">Welcome to our website</p>';
    echo '<div id="mySidenav" class="topnav">';
    echo '<div class="navrow">';
    echo '<div class="navleft">';
    echo '<h2>HowToWebsite</h2>';
    echo '<a href=".."index.php">Home</a>';
    echo '<a class="active" href="templatedot.php">Course</a>';
    echo '<a href="userDashboard.php">Dashboard</a>';
    echo '<a href="discussion.php">Forum</a>';
    echo '<a href="logout.php">Log-out</a>';
    echo '</div>';
    echo '</div>';
    echo '</div>';
}

$currentPage = 'lexidea.php';
$nextPage = 'templatedot.php';
?>
```

Oberer Teil der Webseite geht im Modus eingeloggt, also mit sign-up und log-in Knöpfe sind nicht und anstatt liegt da ein log-out Knopf. Plus die generelle Structur --> Homepage, Cours,...und etc. Knöpfe + Titelen.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href=".."CSS/cours.css">
    <title>How to Website</title>
  </head>
```

Link mit der CSS seite um die Blöcke und andere Styles zu benutzen.


```

<script>

    document.addEventListener('click', function(e) {
        if (e.target.tagName === 'H3' && e.target.parentElement.classList.contains('exercise')) {
            var content = e.target.nextElementSibling;
            toggleDisplay(content);
        }
    });

    function toggleDisplay(element) {
        if (element.style.display === 'block') {
            element.style.display = 'none';
        } else {
            element.style.display = 'block';
        }
    }

</script>

```

Die erste Methode sorgt, ob das richtige Block mit richtigem Titel geklickt wurde, während die 2e Methode das versteckte Block rausweist oder wieder reinmacht.

Die Seitenwechsel-Knöpfe:

```

$currentPage = 'templatedot.php';
$nextPage = 'lexidea.php';

```

Im oberen php-Teil sucht man und gibt an auf welche Seite man sind und welche die ist wo man gehen kann.

```

<div class="card">
    <div class="navigation-buttons">
        <a href="<?php echo $currentPage; ?>" class="button">Previous</a>
        <a href="<?php echo $nextPage; ?>" class="button">Next</a>
    </div>
</div>

```

Ein Block, dass 2 Knöpfe-class hat, die jeweils zurück aktuelle Seite führt oder zu die nächste/vorherige Seite geht.

2 Kurs-Webseiten:

Die 2 differenten Seiten haben jeweils ihren Kurs, die 1e hat den aktuellen Kurs während die 2e die Beispiele Webseiten und Links für Klassen und Erklärungen von Code. Die variablen für die

next/previous Knöpfe müssen auch geändert sein und die 2e hat kein Slide-down also braucht nicht die Methoden. Beide haben dieselben Styles.

User Dashboard:

userDashboard.php

- Es wird wieder eine Sitzung gestartet, um Benutzerdaten zwischen verschiedenen Seitenaufrufen zu speichern und abzurufen. Jedoch mit der Ausnahme, dass man nicht un- eingeloggt ins Dashboard gelangen kann, da man sofort zum login.php gesendet wird mit der header(); Funktion

```

2
3 <?php
4 session_start();
5 if(!isset($_SESSION["user"]))
6 {
7     header("Location: login.php");
8 }
9 else {
10     echo '<input onclick="toggleNav()" type="checkbox" id="nav_check" hidden>';
11     echo '<label for="nav_check" class="hamburger" id="hamburger">';
12     echo '<span>M</span><span>E</span><span>N</span><span>U</span>';
13     echo '</label>';
14     echo '<p class="welcome">Welcome to our website</p>';
15     echo '<div id="mySidenav" class="topnav">';
16     echo '<div class="navrow">';
17     echo '<div class="navleft">';
18     echo '<h2>HowToWebsite</h2>';
19     echo '<a href=" ../index.php">Home</a>';
20     echo '<a href=" ../Includes/templatedot.php">Course</a>';
21     echo '<a class="active" href=" ../Includes/userDashboard.php">Dashboard</a>';
22     echo '<a href=" ../Includes/discussion.php">Forum</a>';
23     echo '<a href=" ../Includes/logout.php">Log-out</a>';
24     echo '</div>';
25     echo '</div>';
26     echo '</div>';
27 }
28

```

Der folgende Code Snippet enthält eine weitere PHP Sektion indem nach den Daten des current Users gesucht werden:

```

29 require_once "database.php";
30 // Fetch user information from the database
31 $username = $_SESSION["user"];
32 $sql = "SELECT dtName, dtEmail, dtDate FROM users WHERE dtName = ?";
33 $stmt = $connection->prepare($sql);
34 $stmt->bind_param("s", $username);
35 $stmt->execute();
36 $result = $stmt->get_result();
37 $user = $result->fetch_assoc();
38
39 if ($user) {
40     $fullName = $user['dtName'];
41     $email = $user['dtEmail'];
42     $creationDate = $user['dtDate'];
43 } else {
44     // Handle the case where user data is not found
45     $fullName = "Unknown";
46     $email = "Unknown";
47     $creationDate = "Unknown";
48 }
49 ?>
50

```

- **Datenbankverbindung:**

```
require_once "database.php";
```

- Hier wird die Datei "database.php" eingebunden, welche vermutlich die Datenbankverbindungsdetails enthält.

- **Benutzername aus der Sitzung holen:**

```
$username = $_SESSION["user"];
```

- Der Benutzername wird aus der Session-Variable `$_SESSION["user"]` geholt.

- **SQL-Abfrage vorbereiten:**

```
$sql = "SELECT dtName, dtEmail, dtDate FROM users WHERE dtName = ?";
```

- Eine SQL-Abfrage wird definiert, um `dtName`, `dtEmail` und `dtDate` aus der Tabelle `users` zu holen, wobei `dtName` mit einem Platzhalter `?` angegeben wird.

- **Abfrage vorbereiten und Parameter binden:**

```
$stmt = $connection->prepare($sql);  
$stmt->bind_param("s", $username);
```

- Die SQL-Abfrage wird vorbereitet und der Platzhalter `?` wird mit dem Benutzernamen (`$username`) als String (`"s"`) ersetzt.

- **Abfrage ausführen:**

```
$stmt->execute();
```

- **Ergebnis abrufen:**

```
$result = $stmt->get_result();  
$user = $result->fetch_assoc();
```

- Das Ergebnis der Abfrage wird geholt und als assoziatives Array (`$user`) gespeichert.

- **Überprüfen, ob Benutzer gefunden wurde:**

```
if ($user) {  
    $fullName = $user['dtName'];  
    $email = $user['dtEmail'];  
    $creationDate = $user['dtDate'];  
} else {  
    $fullName = "Unknown";  
    $email = "Unknown";  
    $creationDate = "Unknown";  
}
```

- Wenn ein Benutzer gefunden wurde, werden die Werte für `fullName`, `email` und `creationDate` aus dem Ergebnis-Array geholt.
- Wenn kein Benutzer gefunden wurde, werden diese Werte auf "Unknown" gesetzt.

Der folgende Code Snippet enthält den html <head> des userDashboard.php files:

```

50
51 <!DOCTYPE html>
52 <html lang="en">
53 <head>
54     <meta charset="UTF-8">
55     <meta name="viewport" content="width=device-width, initial-scale=1.0">
56     <title>HowToWebsite - Dashboard</title>
57     <link rel="stylesheet" href="../CSS/userDashboard.css">
58     <script src="../JS/mobileMenu.js"></script>
59 </head>

```

- **Titel des Dokuments:**

```
<title>HowToWebsite - Dashboard</title>
```

- Der Titel, der im Browser-Tab angezeigt wird, ist "HowToWebsite - Dashboard".

- **CSS-Stylesheet einbinden:**

```
<link rel="stylesheet" href="../CSS/userDashboard.css">
```

- Diese Zeile bindet ein externes CSS-Stylesheet ein, das sich im Verzeichnis ../CSS/ befindet und userDashboard.css heißt. Dieses Stylesheet definiert das Aussehen und Layout der Webseite.

- **JavaScript-Datei einbinden:**

```
<script src="../JS/mobileMenu.js"></script>
```

- Diese Zeile bindet eine externe JavaScript-Datei ein, die sich im Verzeichnis ../JS/ befindet und mobileMenu.js heißt. Dieses Skript könnte Funktionen für das mobile Menü der Webseite enthalten.

Der folgende Code Snippet enthält den ersten Teil des html <body> des userDashboard.php files:

```

60 <body>
61     <main>
62         <div class="canvas">
63             <div class="maintitle">
64                 <h1>Dashboard</h1>
65             </div>
66             <br><br>
67             <div class="row">
68                 <div class="column">
69                     <div class="card" style="display: flex; align-items: center;">
70                         
71                         <h2><?php echo htmlspecialchars($_SESSION["user"]); ?></h2>
72                     </div>
73                     <div class="card">
74                         <ul>
75                             <li><a class="activeButton" href="../Includes/userDashboard.php">Profile Information</a></li>
76                         </ul>
77                     </div>
78                 </div>
79

```

- **Titelbereich:**

```
<div class="maintitle">
  <h1>Dashboard</h1>
</div>
```

- Ein `<div>` mit der Klasse `maintitle`, das ein `<h1>`-Element enthält, welches den Text "Dashboard" anzeigt. Dies ist der Haupttitel der Seite.

- **Reihe und Spalte:**

```
<div class="row">
  <div class="column">
```

- Ein `<div>` mit der Klasse `row`, das eine Reihe definiert, und ein weiteres `<div>` mit der Klasse `column`, das eine Spalte innerhalb dieser Reihe definiert.

- **Karten-Container:**

```
<div class="card" style="display: flex; align-items: center;">
```

- Ein `<div>` mit der Klasse `card`, das eine flexbox-gestützte Karte definiert, die ihre Kinder zentriert ausrichtet.

- **Benutzerbild:**

```

```

- Ein ``-Element, das ein Benutzerbild von `../../Images/ppf/general_pfp.png` lädt. Das Bild hat runde Ecken (`border-radius: 50%`), ist 80x80 Pixel groß und hat einen rechten Außenabstand von 20 Pixeln. Der `alt`-Text ist "user Picture".

- **Benutzername anzeigen:**

```
<h2><?php echo htmlspecialchars($_SESSION["user"]); ?></h2>
```

- Ein `<h2>`-Element, das den Benutzernamen anzeigt. Der Benutzername wird aus der PHP-Session-Variable `$_SESSION["user"]` geholt und mit `htmlspecialchars` gegen XSS-Angriffe gesichert.

- **Weitere Karten:**

```
</div>
<div class="card">
  <ul>
    <li><a class="activeButton" href="../../Includes/userDashboard.php">Profile
    Information</a></li>
  </ul>
</div>
```

- Ein weiteres `<div>` mit der Klasse `card`, das eine ungeordnete Liste (``) enthält. Diese Liste hat einen Listeneintrag (``), der einen Link (`<a>`) mit der Klasse `activeButton` enthält, der auf `../../Includes/userDashboard.php` verweist und den Text "Profile Information" anzeigt.

Der folgende Code Snippet enthält den restlichen html <head> des userDashboard.php files:

```

80         <div class="column">
81             <div class="card">
82                 <h2>Profile Information</h2>
83                 <ul>
84                     <h3>Name:</h3>
85                     <li><?php echo htmlspecialchars($fullName); ?></li>
86                     <br>
87                     <h3>Email:</h3>
88                     <li><?php echo htmlspecialchars($email); ?></li>
89                     <br>
90                     <h3>Date of Creation:</h3>
91                     <li><?php echo htmlspecialchars($creationDate); ?></li>
92                 </ul>
93             </div>
94         </div>
95     </div>
96     <br><br><br><br>
97     <div class="footer">
98         <div class="footer"><p>2GIN - 2023/24</p></div>
99     </div>
100 </div>
101 </main>
102 </body>
103 </html>

```

- Weitere Spalte:

```

• <div class="column">
•     <div class="card">

```

- Im <div> mit der Klasse row, wird eine weitere Spalte mit der Klasse column hinzugefügt und darunter noch ein <div> mit der Klasse card

- Titelbereich:

```
<h2>Profile Information</h2>
```

- Ein <h2>-Element wird angezeigt mit der Aufschrift „Profile Information“

- Liste:

```

• <ul>
•     <h3>Name:</h3>
•     <li><?php echo htmlspecialchars($fullName); ?></li>
•     <br>
•     <h3>Email:</h3>
•     <li><?php echo htmlspecialchars($email); ?></li>
•     <br>
•     <h3>Date of Creation:</h3>
•     <li><?php echo htmlspecialchars($creationDate); ?></li>
• </ul>

```

3 PHP Sektionen sorgen dafür, dass der Name, E-Mail und der Erstellungsdatum des Users angezeigt wird mit Hilfe der Funktion htmlspecialchars();.

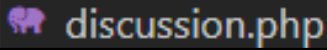
Die Funktion htmlspecialchars() in PHP wird verwendet, um spezielle Zeichen in einer Zeichenkette in HTML-Entitäten zu konvertieren.

Unterhalb des Formulars gibt es noch einen <div> mit der Klasse „footer“, welcher „2GIN – 2023/24“ ganz unten in der Website anzeigt.

```
<div class="footer">  
    <div class="footer"><p>2GIN - 2023/24</p></div>  
</div>
```

Forum:

Discussion.php:



```
<?php
/*Lou */
session_start();
require_once "database.php";

// Check if the user is logged in
if (!isset($_SESSION["user"])) {
    echo ' <p style="margin: 0px; padding-top: 10px; background-color: #1A4D2E; text-align: center; color: white;">Welcome to our website</p>';
    echo ' <div class="topnav">';
    echo ' <div class="row">';
    echo ' <div class="navleft">';
    echo ' <a href="index.php">Home</a>';
    echo ' <a href="templatedot.php">Course</a>';
    echo ' <a href="userDashboard.php">Dashboard</a>';
    echo ' <a class="active" href="discussion.php">Forum</a>';
    echo ' </div>';
    echo ' <div class="navright">';
    echo ' <a href="register.php">Sign up</a>';
    echo ' <a href="login.php">Log in</a>';
    echo ' </div>';
    echo ' </div>';
    echo ' </div>';
} else {
    echo ' <p style="margin: 0px; padding-top: 10px; background-color: #1A4D2E; text-align: center; color: white;">Welcome to our website</p>';
    echo ' <div class="topnav">';
    echo ' <div class="row">';
    echo ' <div class="navleft">';
    echo ' <a href="index.php">Home</a>';
    echo ' <a href="templatedot.php">Course</a>';
    echo ' <a href="userDashboard.php">Dashboard</a>';
    echo ' <a class="active" href="discussion.php">Forum</a>';
    echo ' </div>';
    echo ' <div class="navright">';
    echo ' <a href="logout.php">Log-out</a>';
    echo ' </div>';
    echo ' </div>';
    echo ' </div>';
}
?>
```

Dieser PHP-Code zeigt eine Willkommensnachricht und eine Navigationsleiste auf der Webseite an, die je nach Login-Status des Benutzers unterschiedlich gestaltet ist. Wenn der Benutzer eingeloggt ist, werden andere Links angezeigt, als wenn er nicht eingeloggt ist.

```
session_start();
require_once "database.php";
```

- **Zweck:** Startet eine neue Sitzung oder setzt eine vorhandene Sitzung fort.
- **Funktion:**
 - `session_start()`: Ermöglicht Zugriff auf die Sitzungsvariablen, um Informationen über den Benutzer zu speichern und zu verwalten.

```
if (!isset($_SESSION["user"])) {
    // Code für nicht eingeloggte Benutzer
} else {
    // Code für eingeloggte Benutzer
}
```

- **Zweck:** Überprüft, ob der Benutzer eingeloggt ist.
 - **Mechanismus:**
 - `isset($_SESSION["user"])`: Überprüft, ob die Sitzungsvariable "user" gesetzt ist.
 - `!isset($_SESSION["user"])`: Wenn die Variable nicht gesetzt ist, bedeutet dies, dass der Benutzer nicht eingeloggt ist.
-
- Zeigt eine Willkommensnachricht und Navigationslinks für **nicht eingeloggte Benutzer** an.
 - **Elemente:**
 - **Willkommensnachricht:** `<p style="margin: 0px; padding-top: 10px; background-color: #1A4D2E; text-align: center; color: white;">Welcome to our website</p>`: Zeigt eine Willkommensnachricht in einem grünen Banner an.
 - **Navigationsleiste:** `<div class="topnav">` und `<div class="row">` strukturieren die Navigation.
 - **Link-Bereich:** `<div class="navleft">` enthält Links zu "Home", "Course", "Dashboard" und "Forum".
 - **Benutzeraktionen:** `<div class="navright">` enthält Links zu **"Sign up"** und **"Log in"**.
-
- Zeigt eine Willkommensnachricht und Navigationslinks für eingeloggte Benutzer an.
 - **Elemente:**
 - **Willkommensnachricht:** Wie bei nicht eingeloggten Benutzern.
 - **Navigationsleiste:** Gleiche Struktur wie bei nicht eingeloggten Benutzern.
 - **Link-Bereich:** Gleiche Links zu "Home", "Course", "Dashboard" und "Forum".
 - **Benutzeraktionen:** Anstelle von "Sign up" und "Log in" wird hier **"Log-out"** angezeigt.
-

Header und Kommentarformular:

Zweck: Der HTML-Code stellt den Header mit dem Titel "Forum" dar und zeigt ein Kommentarformular an, das nur für angemeldete Benutzer sichtbar ist.

Mechanismus:

- **Header** (`<header>`): Enthält den Haupttitel "Forum", der die Seite identifiziert.
- **Kommentarformular** (`<form>`): Wird angezeigt, wenn `$_SESSION["user"]` gesetzt ist (d.h., der Benutzer ist angemeldet).
 - Enthält ein verstecktes Feld (`<input type="hidden">`) für die `reply_id`, um auf einen vorhandenen Kommentar zu antworten.
 - Ein Textbereich (`<textarea>`) erlaubt es Benutzern, ihren Kommentar einzugeben.
 - Ein Absenden-Button (`<button>`) ermöglicht das Abschicken des Kommentars.

```
<!--Lou-->
<header>
  <div class="maintitle">
    <h1>Forum</h1>
  </div>
</header>

<br>
<div class="container">
  <?php if (isset($_SESSION["user"])): ?>
    <form action="" method="post">
      <h3 id="title">Leave a Comment</h3>
      <input type="hidden" name="reply_id" id="reply_id">
      <textarea name="comment" placeholder="Your comment" required></textarea>
      <p></p>
      <button class="submit" type="submit" name="submit">Submit</button>
    </form>
  <?php else: ?>
    <p>Please <a href="login.php">log in</a> to leave a comment.</p>
  <?php endif; ?>
</div>

</body>
</html>
```

```
tdocs > 2GIN_HowToWebsite > JS > JS discussionScroll.js > reply
1  function reply(id, name) {
2      const title = document.getElementById('title');
3      title.innerHTML = "Reply to " + name;
4      document.getElementById('reply_id').value = id;
5
6      // Scroll to the top of the page
7      window.scrollTo({ top: 0, behavior: 'smooth' });
8  }
```

JavaScript für die Antwortfunktionalität:

Zweck: Die JavaScript-Funktion `reply()` ändert den Titel des Kommentarformulars und speichert die `reply_id` des angeklickten Kommentars, um darauf zu antworten.

Mechanismus:

- **Funktion `reply()` (JavaScript):**
 - Ändert den Text des `<h3>` Elements mit der ID `title` zu "Antworten auf [Benutzername]".

- Speichert die `reply_id` des angeklickten Kommentars im versteckten Feld `reply_id`.
- Scrollt die Seite nach oben, um das Kommentarformular sichtbar zu machen.

```
<?php
// Function to render nested replies
function renderReplies($reply_id, $connection) {
    $replies = mysqli_query($connection, "SELECT * FROM tb_data WHERE reply_id = $reply_id");
    if (mysqli_num_rows($replies) > 0) {
        echo '<div class="card">';
        echo '<div class="replies">';
        while ($reply_data = mysqli_fetch_assoc($replies)) {
            echo '<div class="reply">';
            echo '<h4>' . htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8') . '</h4>';
            echo '<p>' . htmlspecialchars($reply_data['date'], ENT_QUOTES, 'UTF-8') . '</p>';
            echo '<p>' . htmlspecialchars($reply_data['comment'], ENT_QUOTES, 'UTF-8') . '</p>';
            echo '<button class="reply" onclick="reply(' . $reply_data['id'] . ', \'' . htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8') . '\');">Reply</button>';
            // Recursively render nested replies
            renderReplies($reply_data['id'], $connection);
            echo '</div>';
        }
        echo '</div>';
        echo '</div>';
    }
}
```

discussion.php

Zweck: Die Funktion `renderReplies()` hat den Zweck, alle Antworten (Replies) zu einem bestimmten Kommentar anzuzeigen, indem sie rekursiv die verschachtelten Antworten durchläuft und sie strukturiert auf der Webseite ausgibt.

Mechanismus:

- **Parameter und SQL-Abfrage:** Die Funktion akzeptiert zwei Parameter: `$reply_id` (die ID des Hauptkommentars, zu dem Antworten gesucht werden) und `$connection` (die Datenbankverbindung).

```
function renderReplies($reply_id, $connection) {
    $replies = mysqli_query($connection, "SELECT * FROM tb_data WHERE
    reply_id = $reply_id");
    // ...
}
```

Hier wird eine SQL-Abfrage durchgeführt, um alle Datensätze aus der Tabelle `tb_data` zu selektieren, die die gegebene `reply_id` als `reply_id` haben. Dies holt alle Antworten auf den Hauptkommentar mit der entsprechenden `reply_id`.

- **Datenverarbeitung und Ausgabe:** Die Funktion verarbeitet die abgerufenen Datensätze (Antworten) mit einer Schleife (`while`), die durch das Ergebnis der SQL-Abfrage iteriert.

```
while ($reply_data = mysqli_fetch_assoc($replies)) {
    // ...
}
```

`mysqli_fetch_assoc($replies)`: Diese Funktion ruft den nächsten Datensatz aus dem Ergebnis der SQL-Abfrage (`$replies`) als assoziatives Array (`$reply_data`) ab. Jedes

Array enthält Felder, die den Spalten der Datenbanktabelle entsprechen (z.B. `id`, `username`, `date`, `comment`, usw.).

Ausgabe der Antwort: Innerhalb der Schleife werden die verschiedenen Informationen aus dem `$reply_data`-Array verwendet, um jedes Antwort-Element auf der Webseite anzuzeigen.

```
echo '<div class="reply">';

echo '<h4>' . htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8') . '</h4>';

echo '<p>' . htmlspecialchars($reply_data['date'], ENT_QUOTES, 'UTF-8') . '</p>';

echo '<p>' . htmlspecialchars($reply_data['comment'], ENT_QUOTES, 'UTF-8') . '</p>';
```

`htmlspecialchars()`: Diese Funktion wird verwendet, um sicherzustellen, dass alle in den Kommentaren enthaltenen Daten sicher und korrekt HTML-escaped werden, um XSS-Angriffe zu verhindern.

Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.

Antwort-Button (<button>): Jede Antwort enthält einen Button, der es Benutzern ermöglicht, direkt auf diese spezifische Antwort zu antworten. Der Button ist mit einer JavaScript-Funktion (`reply()`) verknüpft, die es ermöglicht, das Antwortformular vorzubereiten.

```
echo '<button class="reply" onclick="reply(' . $reply_data['id'] . ', \'' . htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8') . '\');">Reply</button>';
```

`'onclick="reply"(...)'`: Diese Zeile ruft die JavaScript-Funktion `reply()` auf, wenn der Benutzer auf den Button klickt. Sie übergibt die `id` des Kommentars, auf den geantwortet wird, sowie den Benutzernamen des Autors dieser Antwort.

- **Rekursive Aufrufe:** Um auch verschachtelte Antworten zu berücksichtigen, ruft die Funktion `renderReplies()` sich selbst rekursiv auf. Dies ermöglicht es, jeden verschachtelten Kommentar auf die gleiche Weise zu verarbeiten und auszugeben.

```
renderReplies($reply_data['id'], $connection);
```

Durch diese rekursive Struktur können beliebig viele Ebenen von Antworten verarbeitet werden, wodurch ein Baumstruktureffekt entsteht, der typisch für Diskussionsforen ist.

- **HTML-Strukturierung:** Die PHP-Ausgabe wird in HTML-Elemente eingebettet, die die visuelle Darstellung der Antworten auf der Webseite definieren. Dies könnte eine Kombination aus `<div>`, `<p>`, `<h4>` und anderen HTML-Tags sein, je nach Designanforderungen.

```

if (isset($_POST["submit"])) {
    if (!isset($_SESSION["user"])) {
        echo '<p>Please log in to post a comment.</p>';
    } else {
        // Prepare the SQL statement with placeholders
        $stmt = $connection->prepare("INSERT INTO tb_data (username, comment, date, reply_id) VALUES (?, ?, ?, ?)");

        // Bind parameters to the prepared statement
        $stmt->bind_param("sssi", $username, $comment, $date, $reply_id);

        // Set the values of parameters
        $username = $_SESSION["user"];
        $comment = $_POST["comment"];
        $date = date('F d Y, h:i:s A');
        $reply_id = $_POST["reply_id"] ?? 0; // Default reply_id to 0 if not set

        // Execute the prepared statement
        if (!$stmt->execute()) {
            die("Error executing query: " . $stmt->error);
        }

        // Redirect to the same page to prevent form resubmission
        header("Location: " . $_SERVER['PHP_SELF']);
        exit;
    }
}

```

Formularkontrolle und Benutzerüberprüfung:

```

if (isset($_POST["submit"])) {
    if (!isset($_SESSION["user"])) {
        echo '<p>Please log in to post a comment.</p>';
    } else {
        // Weiter zur Vorbereitung der SQL-Anweisung
    }
}

```

- **Zweck:** Überprüft, ob das Kommentarformular abgeschickt wurde (`$_POST["submit"]`) und ob der Benutzer angemeldet ist (`$_SESSION["user"]`). Wenn der Benutzer nicht angemeldet ist, wird eine Meldung angezeigt, die ihn auffordert, sich anzumelden.
- **Mechanismus:** Das `isset()`-Statement überprüft die Existenz des `$_POST["submit"]` und `$_SESSION["user"]` Variablen. Wenn das Formular abgeschickt wurde und der Benutzer angemeldet ist, wird der Codeblock unter `else` ausgeführt.

Vorbereitung der SQL-Anweisung:

```

// Prepare the SQL statement with placeholders
$stmt = $connection->prepare("INSERT INTO tb_data (username, comment,
date, reply_id) VALUES (?, ?, ?, ?)");

```

- **Zweck:** Bereitet eine SQL-Anweisung vor, um Daten sicher in die Datenbank einzufügen.
- **Mechanismus:** Die Methode `prepare()` des Verbindungsobjekts (`$connection`) bereitet die SQL-Anweisung vor. Die SQL-Anweisung verwendet Platzhalter (`?`), um Parameter sicher einzubinden.

Parameterbindung an die vorbereitete Anweisung:

```
// Bind parameters to the prepared statement
$stmt->bind_param("sssi", $username, $comment, $date, $reply_id);
```

- **Zweck:** Bindet die Eingabewerte (\$username, \$comment, \$date, \$reply_id) sicher an die vorbereitete SQL-Anweisung. Die Zeichenfolge "sssi" definiert den Typ jedes Parameters (s = string, i = integer).
- **Mechanismus:** Die Methode `bind_param()` des vorbereiteten Statements (\$stmt) bindet die Parameter an die SQL-Anweisung. Dies stellt sicher, dass die Werte sicher in die Datenbank eingefügt werden und SQL-Injektionen vermieden werden.

Setzen der Werte der Parameter:

```
// Set the values of parameters
$username = $_SESSION["user"];
$comment = $_POST["comment"];
$date = date('F d Y, h:i:s A');
$reply_id = $_POST["reply_id"] ?? 0; // Default reply_id to 0 if not set
```

- **Zweck:** Setzt die Werte für die Parameter \$username, \$comment, \$date und \$reply_id basierend auf den Formulardaten (\$_POST) und der aktuellen Zeit (date()).
- **Mechanismus:**
 - `$username = $_SESSION["user"];`: Weist dem \$username die aktuelle angemeldete Benutzer-ID aus der Session zu.
 - `$comment = $_POST["comment"];`: Weist dem \$comment den Wert des abgeschickten Kommentarfelds aus \$_POST zu.
 - `$date = date('F d Y, h:i:s A');`: Setzt \$date auf das aktuelle Datum und die Uhrzeit im angegebenen Format.
 - `$reply_id = $_POST["reply_id"] ?? 0;`: Weist \$reply_id den Wert von \$_POST["reply_id"] zu, falls gesetzt, andernfalls wird der Standardwert 0 verwendet.

Ausführung der vorbereiteten Anweisung:

```
// Execute the prepared statement
if (!$stmt->execute()) {
    die("Error executing query: " . $stmt->error);
}
```

- **Zweck:** Führt die vorbereitete SQL-Anweisung aus, um den Kommentar sicher in der Datenbank zu speichern. Bei einem Fehler wird eine Fehlermeldung ausgegeben und das Skript wird gestoppt.
- **Mechanismus:** Die Methode `execute()` des vorbereiteten Statements (\$stmt) führt die SQL-Anweisung aus, die zuvor vorbereitet und gebunden wurde. Wenn ein Fehler auftritt (`!$stmt->execute()`), wird eine Fehlermeldung mit `die()` ausgegeben und das Skript wird abgebrochen.

Umleitung zur Vermeidung von Formular-Doppelabsendungen:

```
// Redirect to the same page to prevent form resubmission
header("Location: " . $_SERVER['PHP_SELF']);
exit;
```

- **Zweck:** Leitet den Benutzer zur aktuellen Seite zurück (`$_SERVER['PHP_SELF']`), um zu verhindern, dass das Formular erneut abgeschickt wird. `exit` stellt sicher, dass das Skript nach der Umleitung nicht weiterläuft.
- **Mechanismus:** Die Funktion `header()` leitet den Browser zur angegebenen URL weiter (`$_SERVER['PHP_SELF']` ist die aktuelle Skriptdatei). `exit` beendet das Skript, um sicherzustellen, dass nach der Umleitung keine zusätzlichen Aktionen ausgeführt werden.

```
// Fetch and display comments
$datas = mysqli_query($connection, "SELECT * FROM tb_data WHERE reply_id = 0");
while ($data = mysqli_fetch_assoc($datas)) {
    // Pass the connection variable to comment.php
    include 'comment.php';
}
```

Zweck: Dieser PHP-Code dient dazu, alle Hauptkommentare (Kommentare, die keine direkten Antworten sind) aus der Datenbank abzurufen und auf der Webseite anzuzeigen.

Mechanismus:

Datenbankabfrage (`mysqli_query`):

- `mysqli_query($connection, "SELECT * FROM tb_data WHERE reply_id = 0");`
 - Führt eine SQL-Abfrage auf der Datenbanktabelle `tb_data` aus, um alle Datensätze abzurufen, bei denen `reply_id` gleich 0 ist. Dies bedeutet, dass nur Hauptkommentare ausgewählt werden.

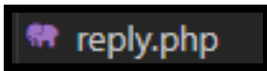
Schleife:

- `while ($data = mysqli_fetch_assoc($datas)) { ... }`
 - Iteriert über jedes Ergebnis der Datenbankabfrage (`$datas`).
 - `mysqli_fetch_assoc($datas)` holt jeden Datensatz als assoziatives Array (`$data`) aus der Abfrageergebnismenge.

Einbinden von `comment.php`:

- `include 'comment.php';`
 - Für jeden Hauptkommentar wird die Datei `comment.php` eingebunden.
 - Diese Datei (`comment.php`) enthält den HTML-Code und ggf. PHP-Code, um einen einzelnen Kommentar auf der Webseite anzuzeigen.

reply.php:



Zweck:

Der Code dient dazu, verschachtelte Antworten auf Kommentare in einem Forum

anzuzeigen. Jede Antwort kann weitere Antworten haben, und diese werden rekursiv angezeigt, um eine hierarchische Struktur zu schaffen.

```
<!--Filipe-->
<?php if (isset($reply_data)): ?>
<div class="reply">
<h4><?php echo htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8'); ?></h4>
<p><?php echo htmlspecialchars($reply_data['date'], ENT_QUOTES, 'UTF-8'); ?></p>
<p><?php echo htmlspecialchars($reply_data['comment'], ENT_QUOTES, 'UTF-8'); ?></p>
<?php $nested_reply_id = $reply_data['id']; ?>
<button class="reply" onclick="reply(<?php echo $nested_reply_id; ?>, '<?php echo htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8'); ?>');">Reply</button>

<?php
// Fetch and display nested replies
$nested_replies = mysqli_query($connection, "SELECT * FROM tb_data WHERE reply_id = $nested_reply_id");
if (mysqli_num_rows($nested_replies) > 0) {
    echo '<div class="replies">';
    while ($nested_reply_data = mysqli_fetch_assoc($nested_replies)) {
        // Include the reply.php to render nested replies
        include 'reply.php';
    }
    echo '</div>';
}
?>
</div>
<?php endif; ?>
```

Überprüfung der Existenz von Antwortdaten (reply_data):

- **Zweck:** Überprüft, ob die Variable `$reply_data` gesetzt ist, bevor die Antwort angezeigt wird.
- **Mechanismus:** Das `if`-Statement stellt sicher, dass nur dann HTML-Code für die Antwort generiert wird, wenn tatsächlich Daten vorhanden sind.

```
<?php if (isset($reply_data)): ?>
```

Anzeige der Antwortdetails:

- **Zweck:** Zeigt die Details der Antwort an, wie den Benutzernamen, das Datum und den Kommentartext.
- **Mechanismus:** Die Funktion `htmlspecialchars()` wird verwendet, um spezielle HTML-Zeichen in ihrer HTML-codierten Form darzustellen. Dies verhindert, dass schädlicher Code, wie JavaScript, in Webseiten eingebettet und ausgeführt wird, was XSS-Angriffe (Cross-Site Scripting) verhindert. Sie ersetzt Zeichen wie `&`, `<`, `>`, `"` und `'` durch ihre sicheren HTML-Entitäten, wodurch Benutzereingaben sicher in HTML eingebettet werden können.

Speichern der reply_id

- **Zweck:** Speichert die ID der aktuellen Antwort in einer Variablen, um sie für weitere verschachtelte Antworten zu verwenden.
- **Mechanismus:** Setzt die Variable `$nested_reply_id` auf die ID der aktuellen Antwort. Dies ermöglicht es, die richtige `reply_id` für verschachtelte Antworten zu verwenden.

```
<?php $nested_reply_id = $reply_data['id']; ?>
```

Antwort-Button

- **Zweck:** Zeigt einen Button an, der es ermöglicht, auf die aktuelle Antwort zu antworten.
- **Mechanismus:** Der Button ruft die JavaScript-Funktion `reply()` auf und übergibt die ID und den Benutzernamen der Antwort als Parameter. Dadurch wird der Kommentarbereich für eine Antwort vorbereitet.

```
<button class="reply" onclick="reply(<?php echo $nested_reply_id; ?>, '<?php echo htmlspecialchars($reply_data['username'], ENT_QUOTES, 'UTF-8'); ?>');">Reply</button>
```

Abrufen und Anzeigen verschachtelter Antworten

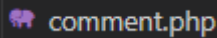
- **Zweck:** Holt alle Antworten, die auf die aktuelle Antwort antworten, aus der Datenbank und zeigt sie an.
- **Mechanismus:** Führt eine SQL-Abfrage aus, um alle Antworten mit der aktuellen `reply_id` zu erhalten. Wenn Antworten vorhanden sind, wird für jede Antwort der gleiche Prozess wiederholt (rekursive Anzeige der verschachtelten Antworten).

```
<?php
// Fetch and display nested replies
$nested_replies = mysqli_query($connection, "SELECT * FROM tb_data WHERE reply_id
= $nested_reply_id");
if (mysqli_num_rows($nested_replies) > 0) {
    echo '<div class="replies">';
    while ($nested_reply_data = mysqli_fetch_assoc($nested_replies)) {
        // Include the reply.php to render nested replies
        include 'reply.php';
    }
    echo '</div>';
}
?>
```

Schließen des Hauptdivs

- **Zweck:** Schließt das `div`-Element, das die gesamte Antwort umgibt.
 - **Mechanismus:** Das schließende `div`-Tag beendet den Antwortblock, wodurch der HTML-Code korrekt geschlossen wird.
-

comment.php:



```
<!--Filipe-->
<link rel="stylesheet" href="../../CSS/comment.css">
<div class="comment">
  <h4><?php echo htmlspecialchars($data['username'], ENT_QUOTES, 'UTF-8'); ?></h4>
  <p><?php echo htmlspecialchars($data['date'], ENT_QUOTES, 'UTF-8'); ?></p>
  <p><?php echo htmlspecialchars($data['comment'], ENT_QUOTES, 'UTF-8'); ?></p>
  <?php $reply_id = $data['id']; ?>
  <button class="reply" onclick="reply(<?php echo $reply_id; ?>, '<?php echo htmlspecialchars($data['username'], ENT_QUOTES, 'UTF-8'); ?>');">Reply</button>

  <?php
    // Call the renderReplies function defined in discussion.php
    renderReplies($reply_id, $connection);
  ?>
</div>
```

Einbinden des CSS-Stylesheets

- **Zweck:** Lädt das Stylesheet für die Kommentare, um das Design und Layout der Kommentarabschnitte zu steuern.
- **Mechanismus:** Der `<link>`-Tag bindet die CSS-Datei `comment.css` aus dem Verzeichnis `../CSS` ein.

Anzeige der Hauptkommentare

- **Zweck:** Zeigt die Details eines Hauptkommentars an, wie den Benutzernamen, das Datum und den Kommentartext.
- **Mechanismus:** Die Funktion `htmlspecialchars()` wird verwendet, um die Daten sicher auszugeben und XSS-Angriffe zu verhindern. Es werden HTML-Tags entschärft, indem Sonderzeichen in HTML-Entities umgewandelt werden.

Speichern der `reply_id`

- **Zweck:** Speichert die ID des aktuellen Kommentars in einer Variablen, um sie für das Antworten auf diesen Kommentar zu verwenden.
- **Mechanismus:** Setzt die Variable `$reply_id` auf die ID des aktuellen Kommentars. Dies ermöglicht es, die richtige `reply_id` für Antworten zu verwenden.

```
<?php $reply_id = $data['id']; ?>
```

Antwort-Button

- **Zweck:** Zeigt einen Button an, der es ermöglicht, auf den aktuellen Kommentar zu antworten.
 - **Mechanismus:** Der Button ruft die JavaScript-Funktion `reply()` auf und übergibt die ID und den Benutzernamen des Kommentars als Parameter. Dadurch wird der Kommentarbereich für eine Antwort vorbereitet.
-

Abrufen und Anzeigen von verschachtelten Antworten

- **Zweck:** Ruft die Funktion `renderReplies` auf, um alle Antworten, die auf den aktuellen Kommentar antworten, aus der Datenbank zu holen und anzuzeigen.
- **Mechanismus:** Die Funktion `renderReplies` wird aufgerufen und erhält die `reply_id` des aktuellen Kommentars sowie die Datenbankverbindung als Parameter. Diese Funktion führt eine SQL-Abfrage aus, um alle Antworten mit der gegebenen `reply_id` abzurufen und rekursiv darzustellen.

Datenbank:

database.php

Dieser PHP-Code stellt eine Verbindung zu einer MySQL-Datenbank her und überprüft, ob die Verbindung erfolgreich war. Wenn die Verbindung fehlschlägt, wird eine Fehlermeldung ausgegeben.

```
<!--Pedro Code-->

<?php
$hostName = "89.58.47.144";
$dbUser = "H2W_User";
$dbPassword = "h2wpw";
$dbName = "dbHow2Website";

// Establish a connection to the database
$connection = mysqli_connect($hostName, $dbUser, $dbPassword, $dbName);

// Check if the connection was successful
if (!$connection) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

- **Zweck:** Definiert die notwendigen Parameter für die Datenbankverbindung.
- **Parameter:**
 - \$hostName: Die IP-Adresse oder der Hostname des Datenbankservers.
 - \$dbUser: Der Benutzername für die Datenbankverbindung.
 - \$dbPassword: Das Passwort für den Datenbankbenutzer.
 - \$dbName: Der Name der Datenbank, zu der die Verbindung hergestellt werden soll.

```
$connection = mysqli_connect($hostName, $dbUser, $dbPassword, $dbName);
```

- **Zweck:** Stellt eine Verbindung zur MySQL-Datenbank her.
- **Funktion:** `mysqli_connect()`
 - Nimmt vier Parameter (Host, Benutzername, Passwort und Datenbankname) und versucht, eine Verbindung zur Datenbank herzustellen.
 - Rückgabe: Liefert ein Verbindungsobjekt zurück, wenn die Verbindung erfolgreich ist, andernfalls `false`.

```
if (!$connection) {
    die("Connection failed: " . mysqli_connect_error());
}
```

- **Zweck:** Überprüft, ob die Verbindung zur Datenbank erfolgreich war.
- **Mechanismus:**
 - `if (!$connection)`: Prüft, ob `$connection false` ist, was bedeutet, dass die Verbindung fehlgeschlagen ist.
 - `die()`: Gibt eine Fehlermeldung aus und beendet das Skript, falls die Verbindung fehlschlägt.
 - `mysqli_connect_error()`: Liefert eine beschreibende Fehlermeldung, warum die Verbindung fehlgeschlagen ist.

4. Verwendete Bibliotheken

Im Laufe der Entwicklung der Website wurde eine Bibliothek zu Nutze gemacht:

```
<link rel="stylesheet" href="https://unpkg.com/98.css">
```

Diese CSS-Bibliothek ermöglicht es Entwicklern, moderne Webanwendungen mit einem nostalgischen Look zu versehen, der an die alten Windows 98-Oberflächen erinnert.

Die CSS-Bibliothek wurde nur zum Designen der Login und Register Seite verwendet

5. Design

Allgemein

Eine Website wird in HTML/PHP geschrieben und durch Styles und CSS wird die Position, Farbe und andere Eigenschaften, der durch Code geschriebenen Objekte, geändert und eingestellt.

Nutzen

Design wird verwendet damit der hardcode Teil der Website nicht einfach links auf weißem Hintergrund, sondern positioniert, mit Hervorhebungen und unterschiedlichen Farben angezeigt wird.

Dies wird gemacht damit man neue Websitennutzer nicht durch "schlechte Grafik" verschreckt und ein allgemein angenehmes Nutzungsklima herrscht.

Wo wurde es implementiert?

Auf allen unseren Seiten, HTML und PHP, wurde eine gewisse Art an Design verwendet.

Manchmal durch externe Bibliotheken, meistens jedoch handgeschrieben.

So wurde die Seite benutzbar.

Nutzen von CSS

Um den Codeanteil pro Seite (in VS-Code) klein zu halten, werden CSS-Dateien verwendet.

Die Seite greift dann im Header Code auf die dazugehörige CSS-Datei zurück und nutzt die darinstehenden Informationen, um die Seite visuell anzupassen.

Wieso ist Design wichtig?

Wie bereits erwähnt ist Design wichtig, um die Nutzererfahrung angenehmer zu machen und alles leicht und farblich erkennbar ist. Ohne ein anständiges Design-Overlay wirkt eine Website unvollständig und unseriös.