CS – Internal Assessment

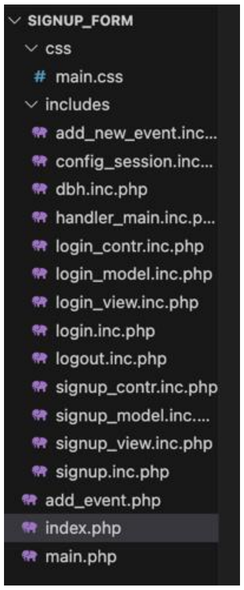
Club Hub

Criterion C – Development

This website is written in php for backend and html with CSS for frontend. It is an informational platform for students and teachers that may have interest in additional activities, such as extracurricular lessons. The program is open for all internet users, has signup/login system with a possibility to add new events to the main page. All the inputted data is stored in the database that is on a hosting service "000webhost.com".

Program functions

List of files needed for the website:



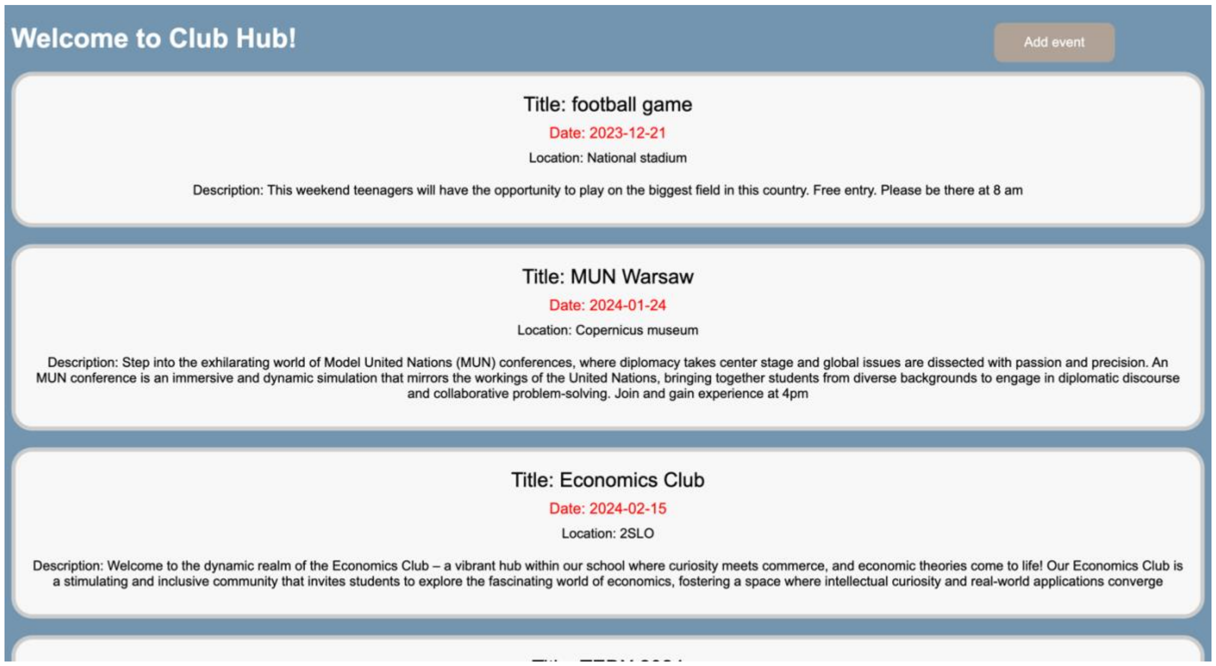
```
SIGNUP_FORM
  css
    main.css
  includes
    add_new_event.inc...
    config_session.inc...
    dbh.inc.php
    handler_main.inc.p...
    login_contr.inc.php
    login_model.inc.php
    login_view.inc.php
    login.inc.php
    logout.inc.php
    signup_contr.inc.php
    signup_model.inc....
    signup_view.inc.php
    signup.inc.php
  add_event.php
  index.php
  main.php
```

The structure is simple: for pages I use ".php". If I need a file for handling something in php, I use ".inc.php" which indicates that these files belong to the folder includes. "_view.inc.php" is for displaying any information. "_contr.inc.php" is for error handling mainly. "_model.inc.php" is for interaction with the database. This project structure shows the modularity, which makes it more reliable and easier to analyse.

List of techniques

- Encryption
- Prepared query
- Loops
- If statements
- Session ID regeneration
- Validation, verification
- Data structures: Arrays
- Modular program
- GUI
- Database

Main page(main.php)



Title: football game

Date: 2023-12-21

Location: National stadium

Description: This weekend teenagers will have the opportunity to play on the biggest field in this country. Free entry. Please be there at 8 am

Title: MUN Warsaw

Date: 2024-01-24

Location: Copernicus museum

Description: Step into the exhilarating world of Model United Nations (MUN) conferences, where diplomacy takes center stage and global issues are dissected with passion and precision. An MUN conference is an immersive and dynamic simulation that mirrors the workings of the United Nations, bringing together students from diverse backgrounds to engage in diplomatic discourse and collaborative problem-solving. Join and gain experience at 4pm

Title: Economics Club

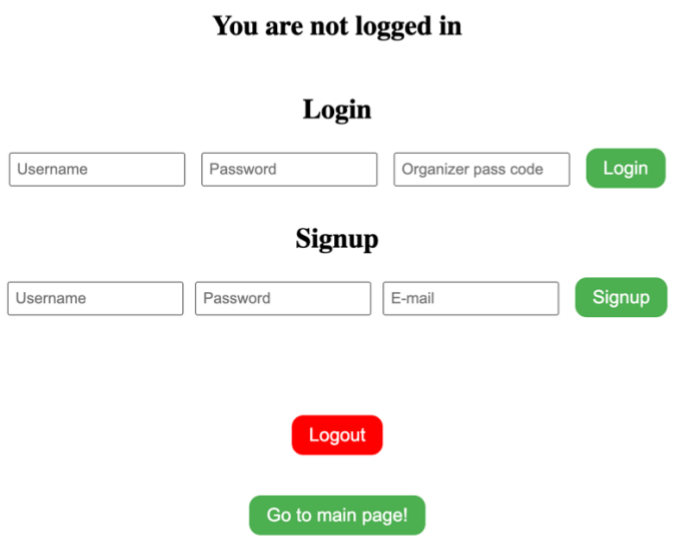
Date: 2024-02-15

Location: 2SLO

Description: Welcome to the dynamic realm of the Economics Club – a vibrant hub within our school where curiosity meets commerce, and economic theories come to life! Our Economics Club is a stimulating and inclusive community that invites students to explore the fascinating world of economics, fostering a space where intellectual curiosity and real-world applications converge

Here every event registered in the database table is displayed in date ascending order for comfort of reviewing. No events with passed date are displayed. They are kept in the database, but users don't need past events.

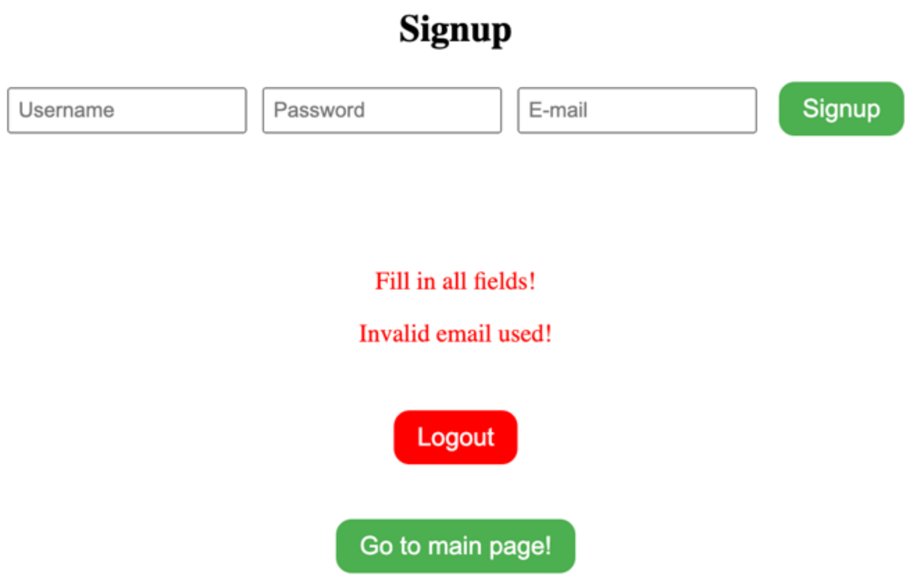
After pressing "Add event" button in the top right corner, the user is transferred to the login/signup page [1][3], since only people who are logged in can add new events:


You are not logged in

Login

Signup

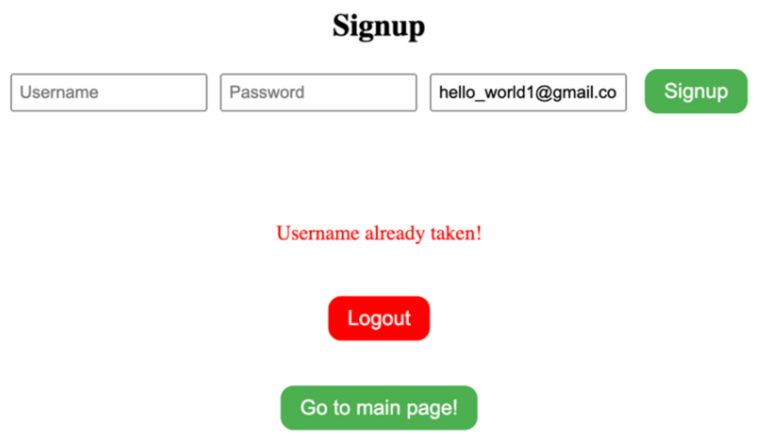
There are a few functions involved to create the best interface for the user (index.php). First, let's consider a case when the user doesn't have an account. Then signup part of the form is needed. With error handling for empty inputs, already existing username and correct email format, a person can create new account(signup):


Signup

Fill in all fields!

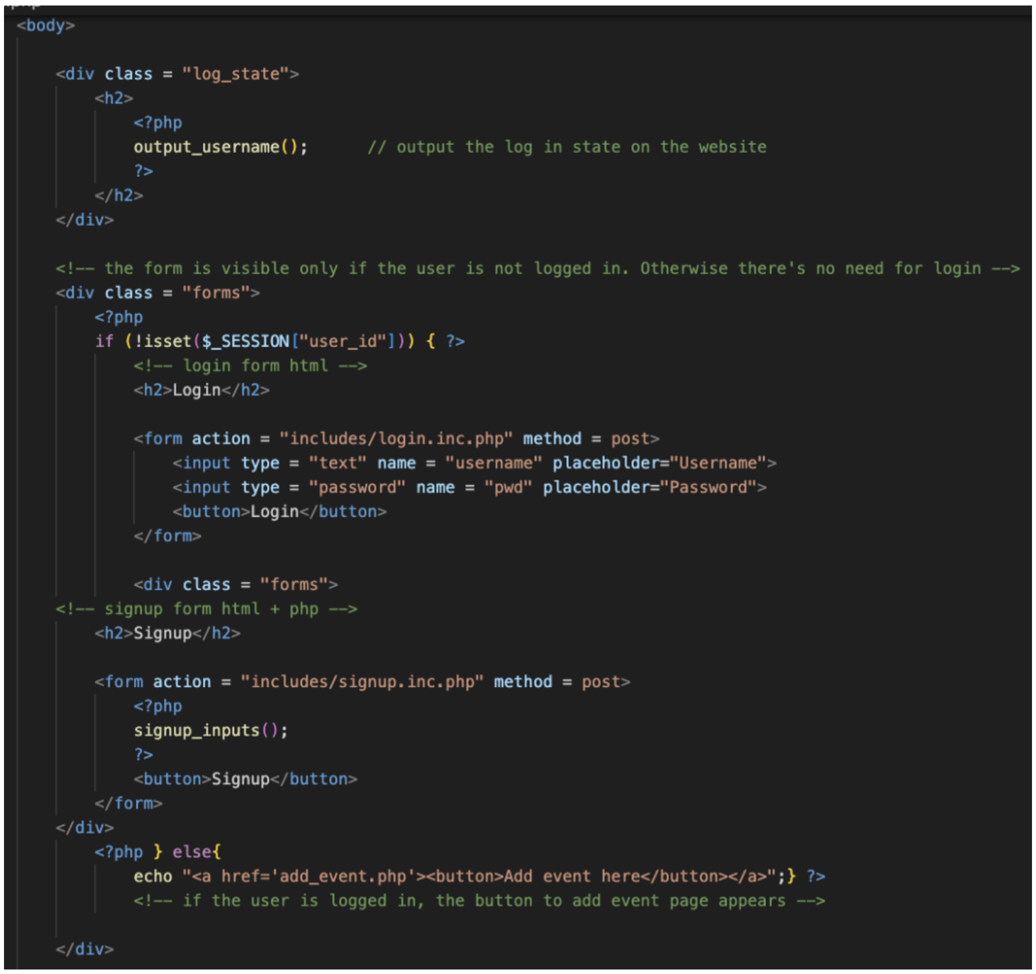
Invalid email used!

Frontend for index.php

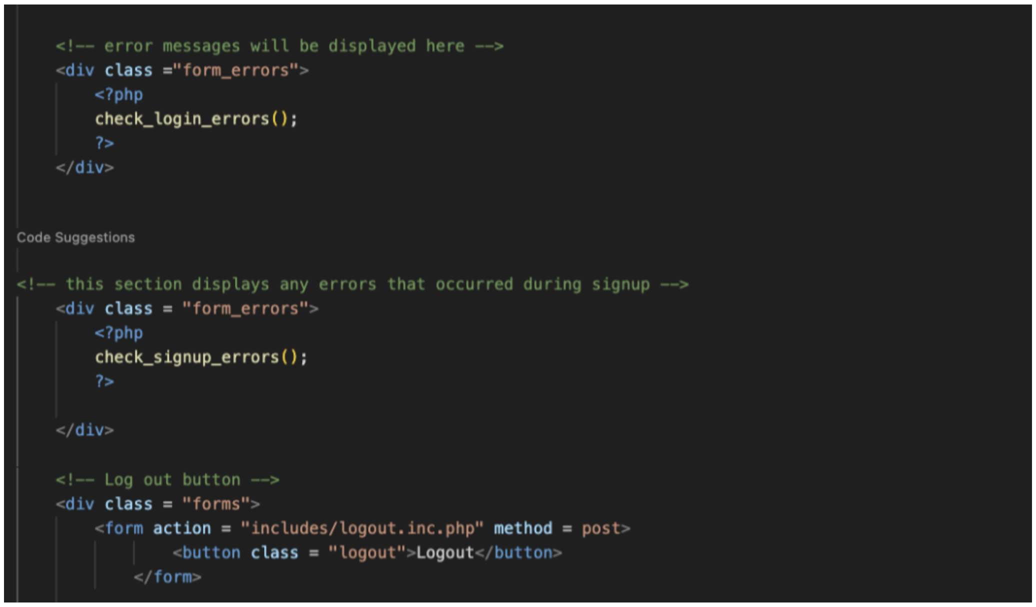


Username already taken!

Notice that the data that is correct, apart from password remains on the page to make the experience of the user more comfortable.



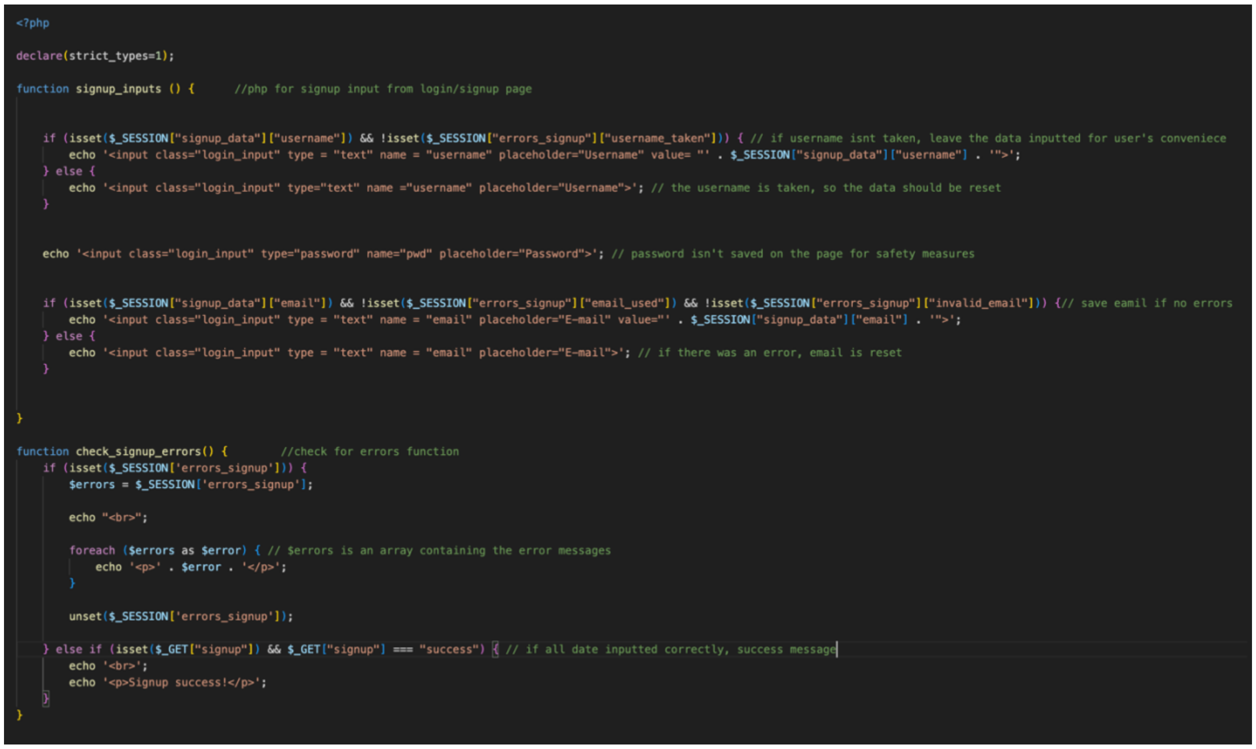
```
<body>

    <div class = "log_state">
        <h2>
            <?php
            output_username();      // output the log in state on the website
            ?>
        </h2>
    </div>

    <!-- the form is visible only if the user is not logged in. Otherwise there's no need for login -->
    <div class = "forms">
        <?php
        if (!isset($_SESSION["user_id"])) { ?>
            <!-- login form html -->
            <h2>Login</h2>

            <form action = "includes/login.inc.php" method = post>
                <input type = "text" name = "username" placeholder="Username">
                <input type = "password" name = "pwd" placeholder="Password">
                <button>Login</button>
            </form>

            <div class = "forms">
    <!-- signup form html + php -->
        <h2>Signup</h2>

        <form action = "includes/signup.inc.php" method = post>
            <?php
            signup_inputs();
            ?>
            <button>Signup</button>
        </form>
    </div>
        <?php } else{
            echo "<a href='add_event.php'><button>Add event here</button></a>";} ?>
            <!-- if the user is logged in, the button to add event page appears -->

    </div>
```



```
    <!-- error messages will be displayed here -->
    <div class ="form_errors">
        <?php
        check_login_errors();
        ?>
    </div>


Code Suggestions

<!-- this section displays any errors that occurred during signup -->
    <div class = "form_errors">
        <?php
        check_signup_errors();
        ?>

    </div>

    <!-- Log out button -->
    <div class = "forms">
        <form action = "includes/logout.inc.php" method = post>
                <button class = "logout">Logout</button>
            </form>
```

Code from index.php

Here are the functions used in the code above:

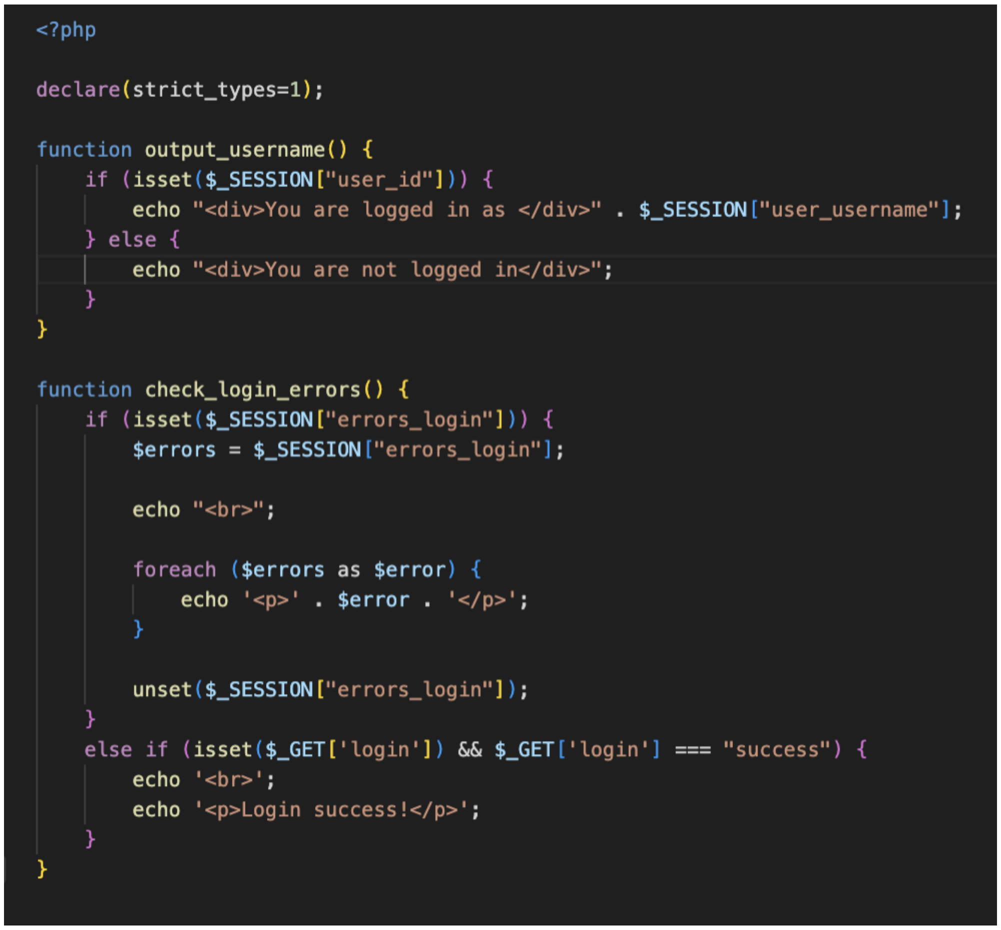


```
<?php

declare(strict_types=1);

function signup_inputs () {      //php for signup input from login/signup page


    if (isset($_SESSION["signup_data"]["username"]) && !isset($_SESSION["errors_signup"]["username_taken"])) { // if username isnt taken, leave the data inputted for user's conveniece
        echo '<input class="login_input" type = "text" name = "username" placeholder="Username" value= "' . $_SESSION["signup_data"]["username"] . '">';
    } else {
        echo '<input class="login_input" type="text" name ="username" placeholder="Username">'; // the username is taken, so the data should be reset
    }


    echo '<input class="login_input" type="password" name="pwd" placeholder="Password">'; // password isn't saved on the page for safety measures


    if (isset($_SESSION["signup_data"]["email"]) && !isset($_SESSION["errors_signup"]["email_used"]) && !isset($_SESSION["errors_signup"]["invalid_email"])) {// save eamil if no errors
        echo '<input class="login_input" type = "text" name = "email" placeholder="E-mail" value="' . $_SESSION["signup_data"]["email"] . '">';
    } else {
        echo '<input class="login_input" type = "text" name = "email" placeholder="E-mail">'; // if there was an error, email is reset
    }


}

function check_signup_errors() {        //check for errors function
    if (isset($_SESSION['errors_signup'])) {
        $errors = $_SESSION['errors_signup'];

        echo "<br>";

        foreach ($errors as $error) { // $errors is an array containing the error messages
            echo '<p>' . $error . '</p>';
        }

        unset($_SESSION['errors_signup']);

    } else if (isset($_GET["signup"]) && $_GET["signup"] === "success") { // if all date inputted correctly, success message
        echo '<br>';
        echo '<p>Signup success!</p>';
    }
}
```

Code from signup_view.inc.php

The part echo "…" . $SESSION["signup_data"]["username"] . (…) and similar to that leave not sensible data on the website to not make the user retype it once again, as described earlier.

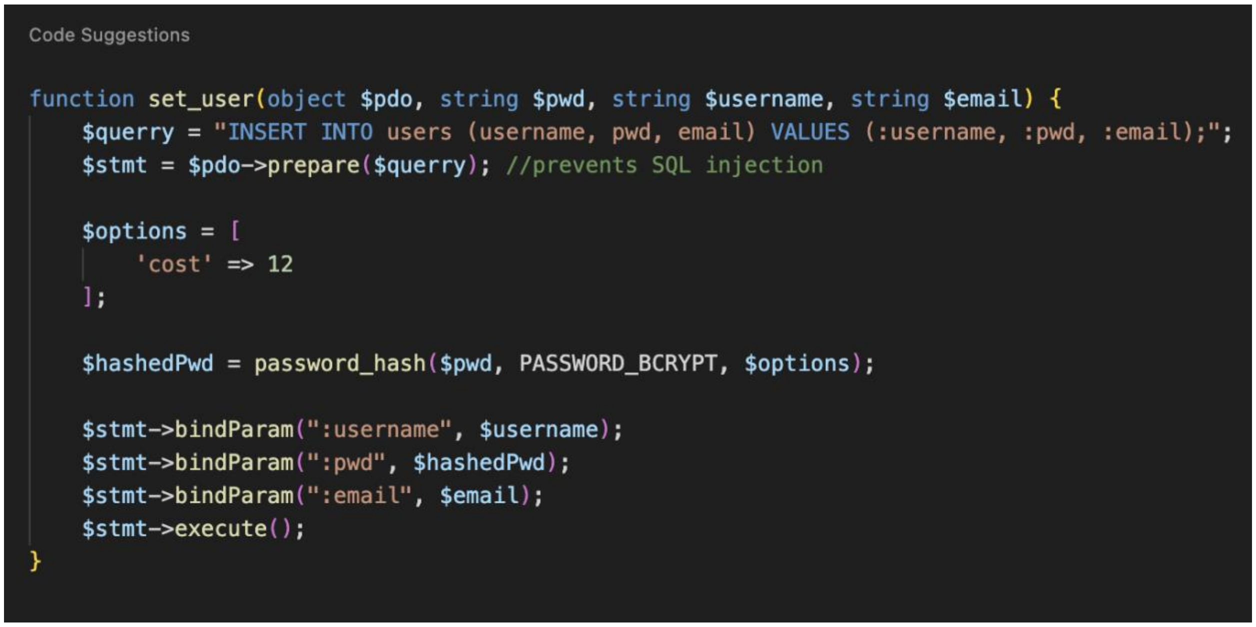
Here the array is used to store all the errors from the session and then the function outputs them below the registration form on the website.



```php
<?php

declare(strict_types=1);

function output_username() {
    if (isset($_SESSION["user_id"])) {
        echo "<div>You are logged in as </div>" . $_SESSION["user_username"];
    } else {
        echo "<div>You are not logged in</div>";
    }
}

function check_login_errors() {
    if (isset($_SESSION["errors_login"])) {
        $errors = $_SESSION["errors_login"];

        echo "<br>";

        foreach ($errors as $error) {
            echo '<p>' . $error . '</p>';
        }

        unset($_SESSION["errors_login"]);
    }
    else if (isset($_GET['login']) && $_GET['login'] === "success") {
        echo '<br>';
        echo '<p>Login success!</p>';
    }
}
```

Code from login_view.inc.php

If everything is successful, the data is inserted into the table:



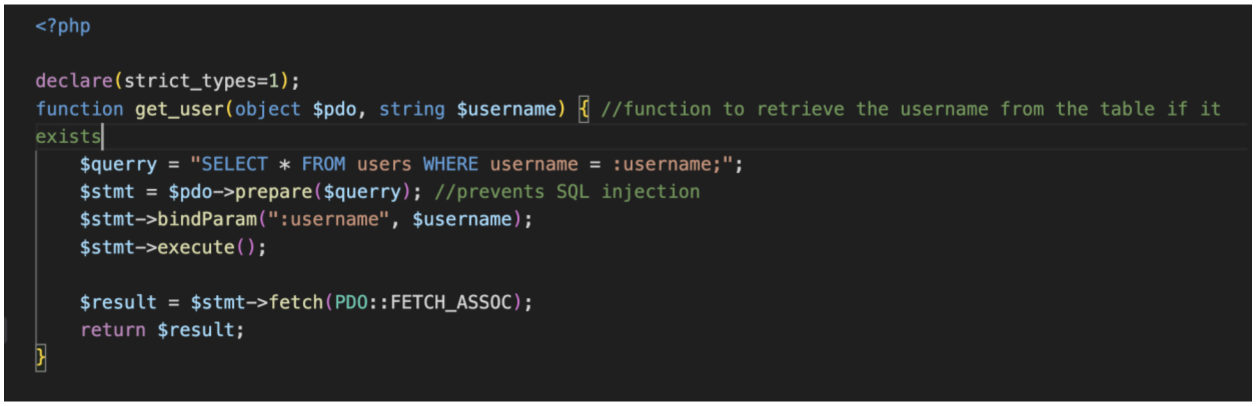
```
Code Suggestions

function set_user(object $pdo, string $pwd, string $username, string $email) {
    $querry = "INSERT INTO users (username, pwd, email) VALUES (:username, :pwd, :email);";
    $stmt = $pdo->prepare($querry); //prevents SQL injection

    $options = [
        'cost' => 12
    ];

    $hashedPwd = password_hash($pwd, PASSWORD_BCRYPT, $options);

    $stmt->bindParam(":username", $username);
    $stmt->bindParam(":pwd", $hashedPwd);
    $stmt->bindParam(":email", $email);
    $stmt->execute();
}
```

I bind the parameter to the value before executing the query because this step prevents someone from inputting an SQL query on the website. That would damage the structure of the table in the database, hence, for safety measures this step is taken.

I wanted to point out here, that inserting the password in the database without encrypting is dangerous for the user's data. It is prone to hacking and leakages. Hence, by using of the hashing technique of php language, I encrypt only the password in 12 symbols in order to ensure security of user's data.

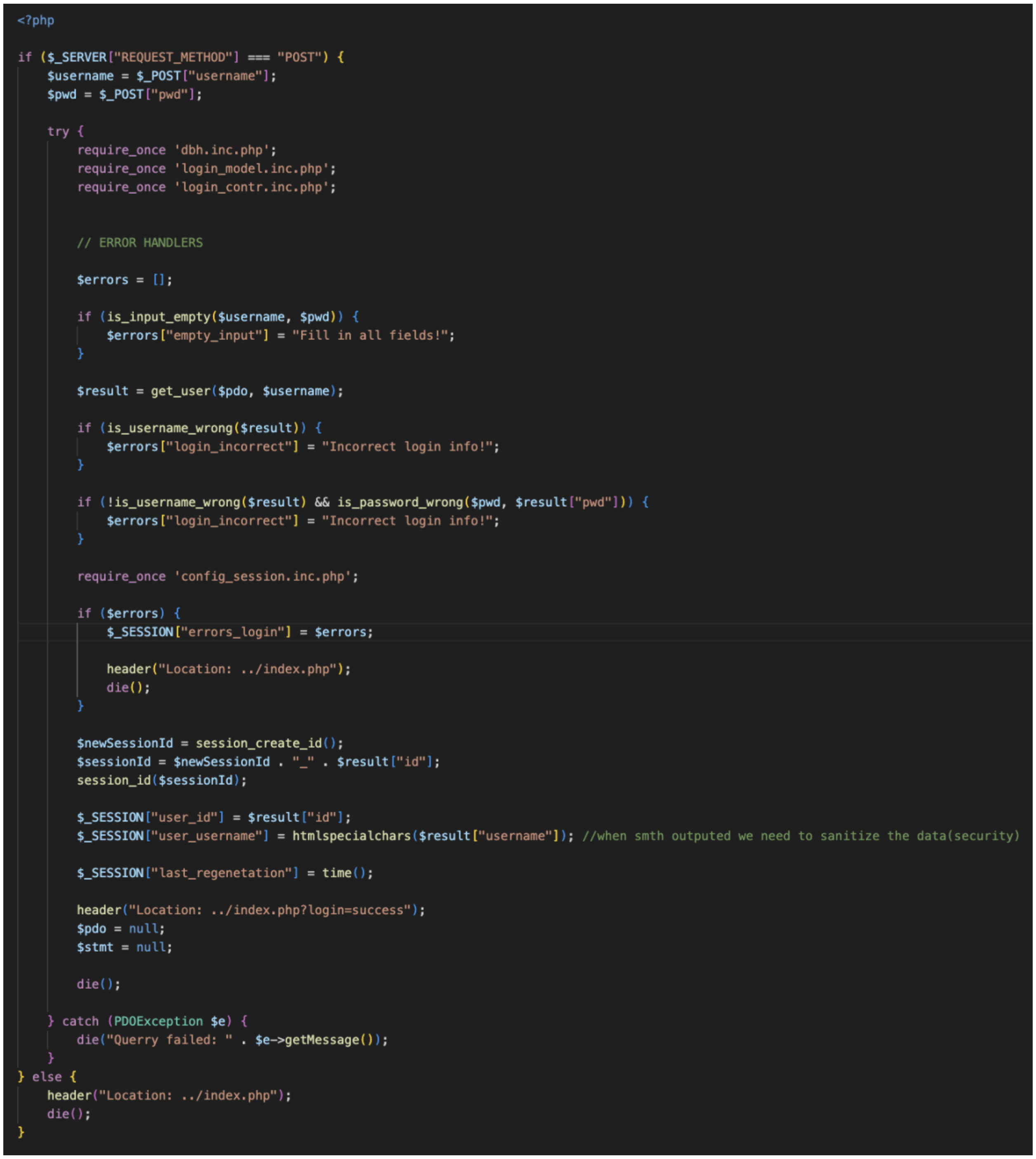
Log in

After creating an account, the user needs to log in with that data. I check for the same errors as in signup part, but here no email field is displayed. In order to log in, the username should be the same as in the table. By using SQL query, I check whether the data is correct:



```
<?php

declare(strict_types=1);
function get_user(object $pdo, string $username) { //function to retrieve the username from the table if it
exists
    $querry = "SELECT * FROM users WHERE username = :username;";
    $stmt = $pdo->prepare($querry); //prevents SQL injection
    $stmt->bindParam(":username", $username);
    $stmt->execute();

    $result = $stmt->fetch(PDO::FETCH_ASSOC);
    return $result;
}
```

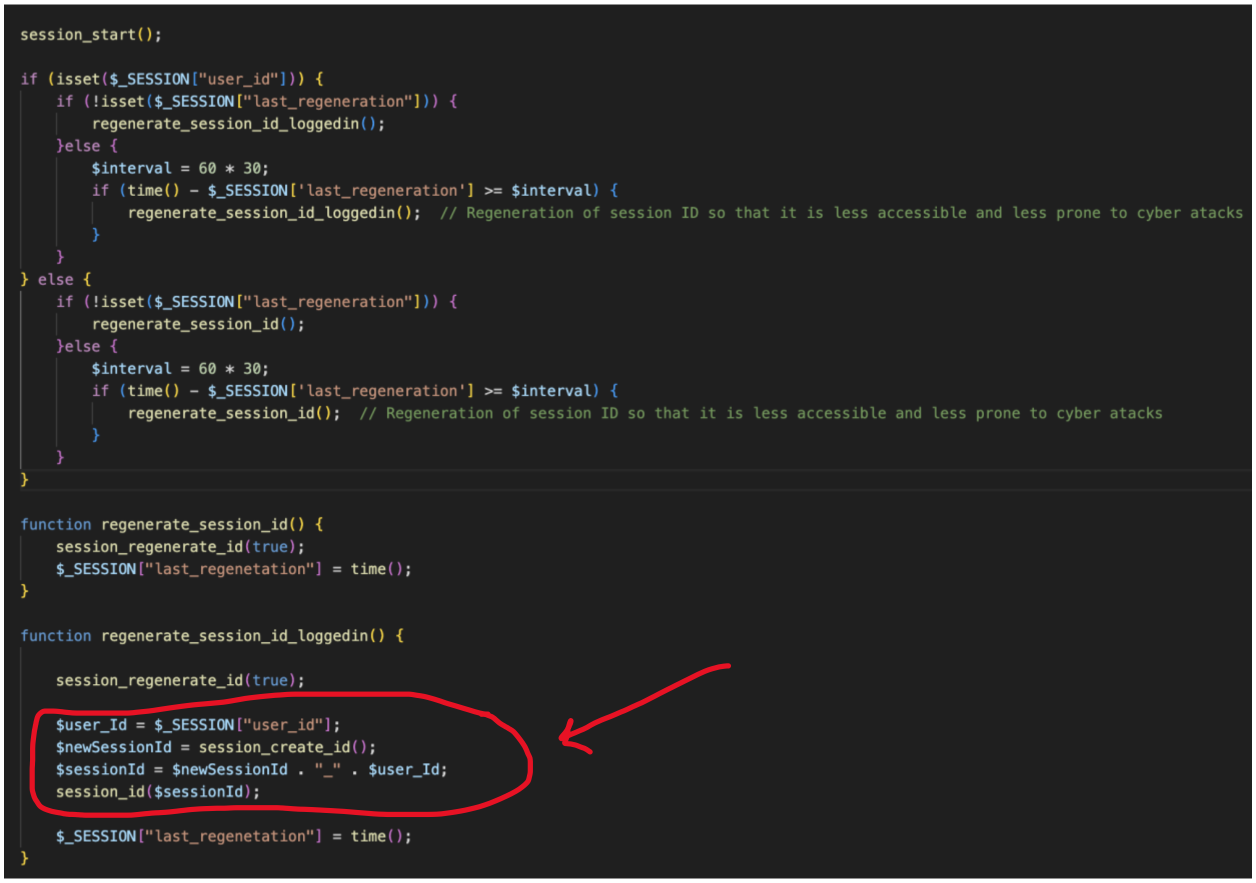
Code from signup_model.inc.php



```php
<?php

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $username = $_POST["username"];
    $pwd = $_POST["pwd"];

    try {
        require_once 'dbh.inc.php';
        require_once 'login_model.inc.php';
        require_once 'login_contr.inc.php';


        // ERROR HANDLERS

        $errors = [];

        if (is_input_empty($username, $pwd)) {
            $errors["empty_input"] = "Fill in all fields!";
        }

        $result = get_user($pdo, $username);

        if (is_username_wrong($result)) {
            $errors["login_incorrect"] = "Incorrect login info!";
        }

        if (!is_username_wrong($result) && is_password_wrong($pwd, $result["pwd"])) {
            $errors["login_incorrect"] = "Incorrect login info!";
        }

        require_once 'config_session.inc.php';

        if ($errors) {
            $_SESSION["errors_login"] = $errors;

            header("Location: ../index.php");
            die();
        }

        $newSessionId = session_create_id();
        $sessionId = $newSessionId . "_" . $result["id"];
        session_id($sessionId);

        $_SESSION["user_id"] = $result["id"];
        $_SESSION["user_username"] = htmlspecialchars($result["username"]); //when smth outputed we need to sanitize the data(security)

        $_SESSION["last_regenetation"] = time();

        header("Location: ../index.php?login=success");
        $pdo = null;
        $stmt = null;

        die();

    } catch (PDOException $e) {
        die("Querry failed: " . $e->getMessage());
    }
} else {
    header("Location: ../index.php");
    die();
}
```

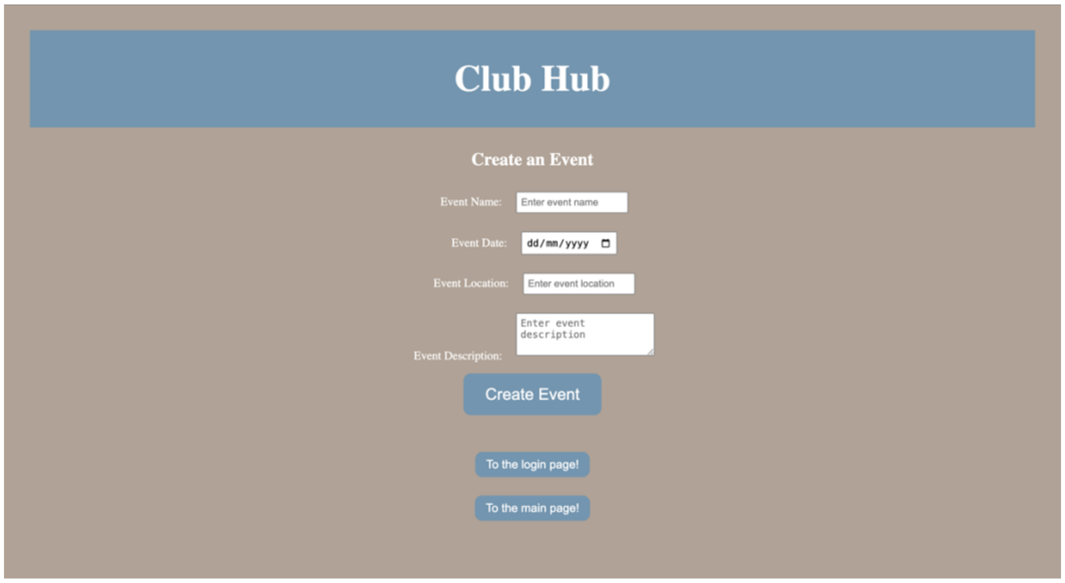
The first part of the code above deals with the errors that may occur. The second half is aimed to regenerate the id of the current session in order to decrease the chance of getting hacked and data leakage.

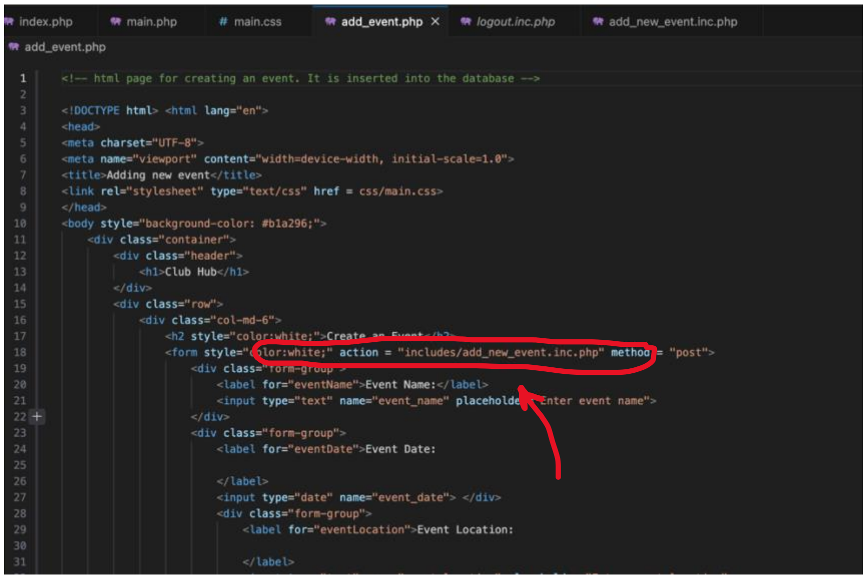
In this part of the code, I generate new session ID as mentioned previously to increase security of the data still on the page. The arrow shows that when the user is logged in, session ID is regenerated once again and modified with user_id. It is a primary key in the table assigned to each user. Incremented for every entry by one.



```
session_start();

if (isset($_SESSION["user_id"])) {
    if (!isset($_SESSION["last_regeneration"])) {
        regenerate_session_id_loggedin();
    }else {
        $interval = 60 * 30;
        if (time() - $_SESSION['last_regeneration'] >= $interval) {
            regenerate_session_id_loggedin();  // Regeneration of session ID so that it is less accessible and less prone to cyber atacks
        }
    }
} else {
    if (!isset($_SESSION["last_regeneration"])) {
        regenerate_session_id();
    }else {
        $interval = 60 * 30;
        if (time() - $_SESSION['last_regeneration'] >= $interval) {
            regenerate_session_id();  // Regeneration of session ID so that it is less accessible and less prone to cyber atacks
        }
    }
}

function regenerate_session_id() {
    session_regenerate_id(true);
    $_SESSION["last_regenetation"] = time();
}

function regenerate_session_id_loggedin() {

    session_regenerate_id(true);

    $user_Id = $_SESSION["user_id"];
    $newSessionId = session_create_id();
    $sessionId = $newSessionId . "_" . $user_Id;
    session_id($sessionId);

    $_SESSION["last_regenetation"] = time();
}
```

After dealing with the login/signup form, user has an option to add a new event. The person is transferred to the “add event” page:

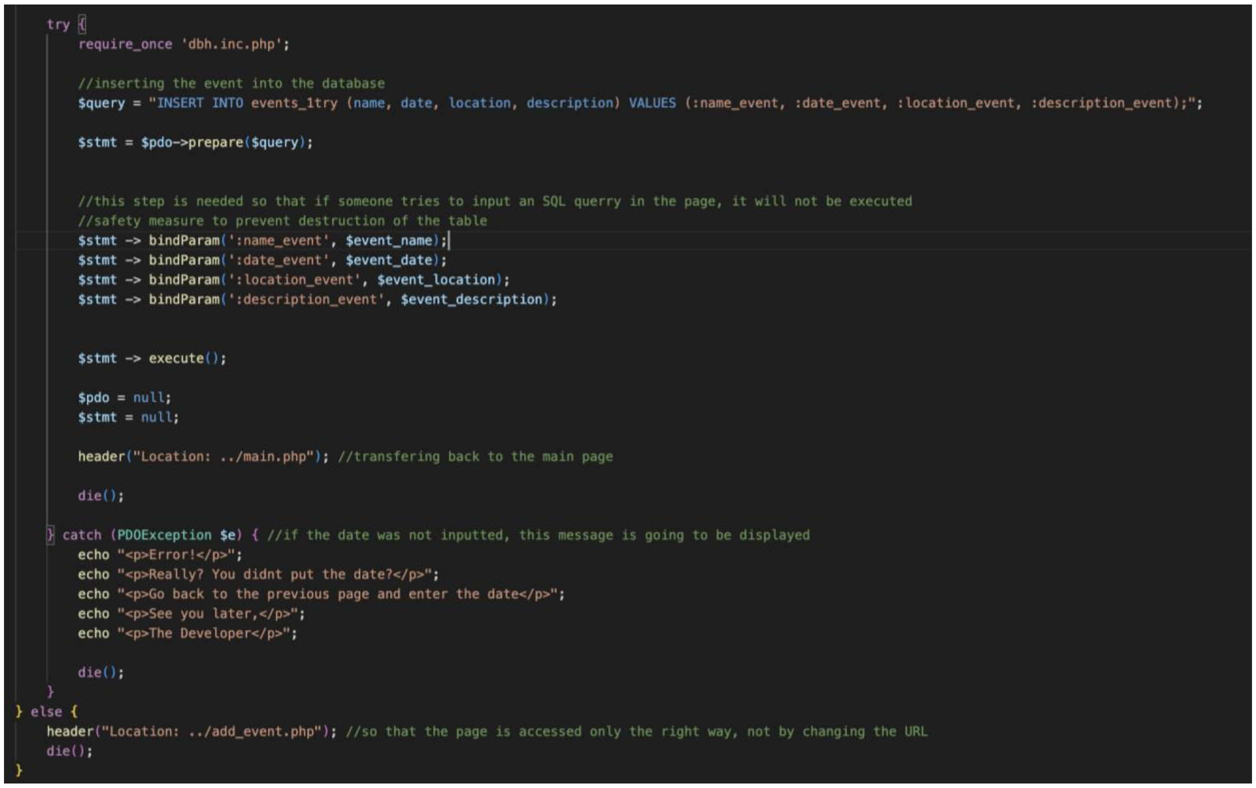




```
<!-- html page for creating an event. It is inserted into the database -->

<!DOCTYPE html> <html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Adding new event</title>
<link rel="stylesheet" type="text/css" href = css/main.css>
</head>
<body style="background-color: #b1a296;">
    <div class="container">
        <div class="header">
            <h1>Club Hub</h1>
        </div>
        <div class="row">
            <div class="col-md-6">
                <h2 style="color:white;">Create an Event</h2>
                <form style="color:white;" action = "includes/add_new_event.inc.php" method = "post">
                    <div class="form-group">
                        <label for="eventName">Event Name:</label>
                        <input type="text" name="event_name" placeholder="Enter event name">
                    </div>
                    <div class="form-group">
                        <label for="eventDate">Event Date:

                        </label>
                        <input type="date" name="event_date"> </div>
                        <div class="form-group">
                            <label for="eventLocation">Event Location:

                            </label>
```

Code from add_event.php [2], [4]

Here the user can input the information about the event. The code for the page is purely html, but with the action of the form the information is sent to the processing file add_new_event.inc.php (red arrow).

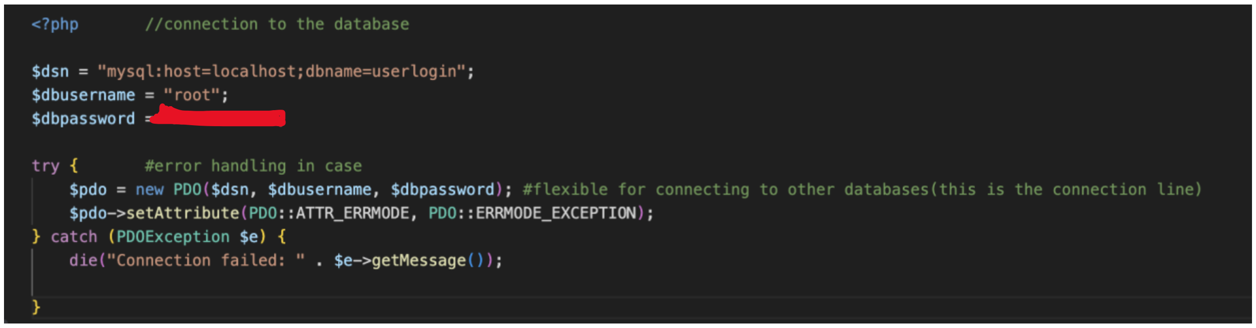


```php
    try {
        require_once 'dbh.inc.php';

        //inserting the event into the database
        $query = "INSERT INTO events_1try (name, date, location, description) VALUES (:name_event, :date_event, :location_event, :description_event);";

        $stmt = $pdo->prepare($query);


        //this step is needed so that if someone tries to input an SQL querry in the page, it will not be executed
        //safety measure to prevent destruction of the table
        $stmt -> bindParam(':name_event', $event_name);
        $stmt -> bindParam(':date_event', $event_date);
        $stmt -> bindParam(':location_event', $event_location);
        $stmt -> bindParam(':description_event', $event_description);


        $stmt -> execute();

        $pdo = null;
        $stmt = null;

        header("Location: ../main.php"); //transfering back to the main page

        die();

    } catch (PDOException $e) { //if the date was not inputted, this message is going to be displayed
        echo "<p>Error!</p>";
        echo "<p>Really? You didnt put the date?</p>";
        echo "<p>Go back to the previous page and enter the date</p>";
        echo "<p>See you later,</p>";
        echo "<p>The Developer</p>";

        die();
    }
} else {
    header("Location: ../add_event.php"); //so that the page is accessed only the right way, not by changing the URL
    die();
}
```

Code from add_new_event.inc.php

In each file I used a try-catch block, which resembles an if statement. It is often utilized for errors as well, for example if something doesn't happen – execute catch part. The code is responsible for all the actions taken when the user creates the event on "add_event.php" page. The comments describe the processes. Here I use parameter binding once again, and in the case of not inputting the date of the event, the error is displayed in the catch block at the bottom.

The database is used for storing user data, events, and comments. The connection to the DB is done in the separate file, which is requested when actions are to be done, for example inserting into the DB.



```php
<?php       //connection to the database

$dsn = "mysql:host=localhost;dbname=userlogin";
$dbusername = "root";
$dbpassword = [redacted]

try {       #error handling in case
    $pdo = new PDO($dsn, $dbusername, $dbpassword); #flexible for connecting to other databases(this is the connection line)
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());

}
```

Word count: 870

References:

1. Php tutorial - https://youtu.be/yrFr5PMdk2A?si=-MOsE5Z6s0Q81rLM
2. Html guide - https://www.w3schools.com/html/html_symbols.asp
3. Php guide - https://youtu.be/pWG7ajC_OVo?si=oCQUvGYyFgtKK3Iw
4. CSS guide - https://www.w3schools.com/cssref/pr_padding-top.php