

Data.php

```
<?php
//Loading classes
spl_autoload_register('autoload');
function autoload($sClassname): void
{
    require_once($sClassname . ".php");
}
//Load pre-defined users and add to UserList
$user = new User("admin1", "abcdefgh", "email", 1); //ADMIN
$user1 = new User("PhpLover1", "abcdefgh", "email"); //USER
$user2 = new User("jdoe", "abcdefgh", "email"); //USER
$userList = new UserList(); //VERWALTUNG ALLER USER
$userList->addUser($user);
$userList->addUser($user1);
$userList->addUser($user2);

//An admin is logged in in the recent TodoList
$todoList = new TodoList($user);

//Creating various tasks
$entry = new TodoListItem("e1", new DateTime("2021-02-01"), $user2->userId,
"Hausarbeit", new DateTime("2021-02-01"));
$entry1 = new TodoListItem("e2", new DateTime("2021-04-05"), $user1-
>userId, "Hausübung machen", new DateTime("2021-04-02"), "Dringend die PHP
Hausübung abgeben :D");
$entry2 = new TodoListItem("e3", new DateTime("2021-03-06"), $user1-
>userId, "Yoga", new DateTime("2021-03-02"));
$entry3 = new TodoListItem("e4", new DateTime("2021-08-05"), $user2-
>userId, "Prokrastinieren", new DateTime("2021-03-02"), "Ich bin eine
Notiz, hallo!");

//Add tasks to TodoList
$todoList->addEntry($entry);
$todoList->addEntry($entry1);
$todoList->addEntry($entry2);
$todoList->addEntry($entry3);
?>
```

Index.php

```
<?php
require_once "data.php";
session_start(); //Zugriff auf Session Array
?>

<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ToDoList</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.
css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
</head>
<body>

<?php
global $todoList; //ToDoList of data.php
global $userList; //UserList of data.php
//Vordefinierte Todoliste soll nur beim ersten Mal eingespielt werden
//Ermöglicht Aktualisieren der Todoliste
//und der Userliste mittels Session-Variable

//Pre-defined ToDoList should only be loaded at the first time
//So the Todolist can be updated frequently with new tasks
//and wont be overwritten by the old ToDoList
if(!isset($_SESSION["todolist"])){
    $_SESSION["todolist"] = $todoList;
}

//Pre-defined UserList should only be loaded at the first time
//So the UserList can be updated frequently with new users
//and wont be overwritten by the old UserList
else if(!isset($_SESSION["userlist"])){
    $_SESSION["userlist"] = $userList;
}
//Login and Registration Process incl. Error-Handling
$errors = [];
if(!isset($_SESSION["user"])){

    //case user is not logged in
    if(isset($_REQUEST["action"]) && $_REQUEST["action"] == "login"){

        //case user wants to log in
        if(isset($_REQUEST["username"])
            && isset($_REQUEST["password"])
            && $_REQUEST["username"] != ""
            && $_REQUEST["password"] != ""
            && checkLogin($_REQUEST["username"], $_REQUEST["password"])){
```

```

        //case user data is correct
        $_SESSION["user"] = $_REQUEST["username"];
        showData();
        showTaskForm();
        showLogoutForm();
    }
    else{
        //case user data is incorrect
        echo("<div class='notify red'>Login data incorrect!</div>");
        showLoginForm();
        showRegistrationForm();
    }
}
else if(isset($_REQUEST["action"]) && $_REQUEST["action"] ==
"register"){

    //case user wants to register
    if(checkRegistration()){

        //data correct
        register();
    }
    else{
        //data incorrect
        echo("<div class='notify red'>Registration data not
correct!</div>");
    }
    showLoginForm();
    showRegistrationForm();
}
else{
    //case user wants to view page
    showLoginForm();
    showRegistrationForm();
}
}
//case user is logged in
else{
    //case user creates new task
    if(isset($_REQUEST["action"]) && $_REQUEST["action"] == "addTasks"){
        if(checkAddEntry())
            actualizeData();
        showData();
        showTaskForm();
        showLogoutForm();
    }
    //case user wants to logout
    else if(isset($_REQUEST["action"]) && $_REQUEST["action"] == "logout"){
        unset($_SESSION["user"]);
        showLoginForm();
        showRegistrationForm();
    }
    //case user wants to view page
    else{
        showData();
        showTaskForm();
        showLogoutForm();
    }
}
}

```

```

//register()
//This function allows a new user to register by entering all
//data needed. If the user wants to register with a already
//existing username, a error message shows up.
function register(){
    try {
        $newUser = new User($_REQUEST["username"],
$_REQUEST["password"], $_REQUEST["email"]);
        $_SESSION["userlist"]->addUser($newUser);
        echo("<div class='notify green'>Registration successful!</div>");
    } catch (Exception $e){
        echo("<div class='notify red'>". $e->getMessage(). "</div>");
    }
}

//actualizeData()
//The TodoList will be updated by adding a new TodoListItem.
function actualizeData(){
    $entry = new TodoListItem("e".rand(), new DateTime("now"),
$_SESSION["user"], $_REQUEST["addTitle"], new DateTime("now"),
$_REQUEST["addNote"]);
    $_SESSION["todolist"]->addEntry($entry);
}

//checkAddEntry()
//Checks the input fields of the form to add new tasks
function checkAddEntry():bool{
    global $errors;
    if(!isset($_REQUEST["addTitle"]) || strlen($_REQUEST["addTitle"]) < 6)
        $errors["addTitle"] = "<i>Title must be at least 6 characters
long!</i>";

    if(count($errors) > 0)
        return false;
    return true;
}

//checkLogin()
//Checks the input fields of the login form by comparing it with the saved
data of the userList.
//Only if username and password are correct, the user can login.
function checkLogin(string $username, string $password):bool{
    return($username == $_SESSION["userlist"]->users[$username]->userId &&
$password == $_SESSION["userlist"]->users[$username]->password);
}

//checkRegistration()
//Checks if various rules of the input fields of the registration form
//are followed. Otherwise the errors will be saved.
function checkRegistration():bool{
    global $errors;
    //username
    if(!isset($_REQUEST["username"]) || strlen($_REQUEST["username"])
< 6)
        $errors["username"] = "<br><i>Username must be at least 6
characters long!</i>";

```

```

//password
if(!isset($_REQUEST["password1"]) ||
    !isset($_REQUEST["password2"]) ||
    $_REQUEST["password1"] != $_REQUEST["password2"] ||
    strlen($_REQUEST["password1"]) < 8)
    $errors["password1"] = "<br><i>Password must be at least 8
characters long and must match</i>";

//email
if(!isset($_REQUEST["email"]) || filter_var($_REQUEST["email"],
FILTER_VALIDATE_EMAIL) === false)
    $errors["email"] = "<br><i>Must be a valid email address</i>";

if(count($errors) > 0)
    return false;
return true;
}

//showData()
//This function prints the (updated) TodoList. Only tasks of the
//user who is logged in will be shown, all other tasks wont be
//visible for this user.
function showData(){
    $user = $_SESSION["userlist"]->users[$_SESSION["user"]];
    $_SESSION["todolist"]->loginUser = $user;
    echo($_SESSION["todolist"]);
}

//showTaskForm()
//Output of the TaskForm - if errors occur, the incorrect fields
//will be highlighted and the error will be described
function showTaskForm(){
    global $errors;
    $title = "";

    if(isset($_REQUEST["addTitle"]) && count($errors) > 0){
        $title = $_REQUEST["addTitle"];
    }
    ?>

    <div class="form-style-5">
        <form action="php echo($_SERVER['PHP_SELF']);?&gt;" method="post"&gt;
            &lt;fieldset&gt;
                &lt;legend&gt;&lt;span class="number"&gt;+&lt;/span&gt;Eintrag
hinzufügen&lt;/legend&gt;
                &lt;?php
                    if(isset($errors["addTitle"])){
                        echo('&lt;input type="text" name="addTitle" id="addTitle"
placeholder="Titel*" class="error" value="'. $title. '"&gt;');
                        echo($errors["addTitle"]);
                    }
                    else echo('&lt;input type="text" name="addTitle" id="addTitle"
placeholder="Titel*" value="'. $title. '"&gt;');

                    if(isset($_REQUEST["addNote"]) &amp;&amp; count($errors) &gt; 0){
                        $desc = $_REQUEST["addNote"];
                        echo('&lt;textarea name="addNote"
id="addNote"&gt;'. $desc. '&lt;/textarea&gt;');
                    }
</pre

```

```

        else echo('<textarea name="addNote" id="addNote"
placeholder="Beschreibung"></textarea>');
    ?>
</fieldset>
<input type="hidden" name="action" value="addTasks">
<input type="submit" id="addTasks" value="Eintrag hinzufügen">
</form>
</div>
<?php
}

//showRegistrationForm()
//Output of the registration form - if errors occur, the incorrect fields
//will be highlighted and the error will be described
function showRegistrationForm() {
    global $errors;
    $usernameVal = "";
    $emailVal = "";

    if(isset($_REQUEST["usernamereg"]) && count($errors) > 0) {
        $usernameVal = $_REQUEST["usernamereg"];
    }
    if(isset($_REQUEST["email"]) && count($errors) > 0) {
        $emailVal = $_REQUEST["email"];
    }
    ?>
    <div class="registration_form">
        <h3>Register:</h3>
        <form action="<?php echo($_SERVER['PHP_SELF']);?>" method="post">

            <label for="usernamereg">Username:</label>
            <?php
                if(isset($errors["usernamereg"])) {
                    echo('<input type="text" name="usernamereg"
id="usernamereg" value="'. $usernameVal. '" class="error">');
                    echo($errors["usernamereg"]);
                }
                else
                    echo('<input type="text" name="usernamereg"
id="usernamereg" value="'. $usernameVal. '">');
            ?>
            <br>

            <label for="password1">Password:</label>
            <?php
                if(isset($errors["password1"])) {
                    echo('<input type="password" name="password1"
id="password1" class="error">');
                    echo($errors["password1"]);
                }
                else
                    echo('<input type="password" name="password1"
id="password1">');
            ?>
            <br>

            <label for="password2">Repeat:</label>
            <input type="password" name="password2" id="password2"><br>
            <label for="email">Email:</label>
            <?php

```

```

        if(isset($errors["email"])){
            echo('<input type="email" name="email" id="email"
class="error" value="'. $emailVal. '">');
            echo($errors["email"]);
        }
        else echo('<input type="email" name="email" id="email"
value="'. $emailVal. '">');
        ?>
        <br>
        <input type="hidden" name="action" value="register">
        <input type="submit" value="Register" class="reg_button">
    </form>
</div>
<?php
}

//showLoginForm()
//Output of the login form
function showLoginForm(){?>
    <div class="login_form">
    <h3>Login:</h3>
    <form action="<?php echo($_SERVER['PHP_SELF']);?>" method="post">
        <label for="username">Username:</label>
        <input type="text" name="username" id="username"/><br>
        <label for="password">Password:</label>
        <input type="password" name="password" id="password"/><br>
        <input type="hidden" name="action" value="login">
        <input type="submit" class="login_button" value="Login">
    </form>
    </div><?php
}

//showLogoutForm()
//Output of the logout button
function showLogoutForm(){ ?>
    <form action="<?php echo($_SERVER['PHP_SELF']);?>" method="post">
        <input type="hidden" name="action" value="logout">
        <input type="submit" value="Logout" class="logout_button">
    </form>
    <?php
}?>
</body>
</html>

```

ToDoList.php

```
<?php

class ToDoList
{
    public function __construct(
        private User $loginUser,
        private array $entries = []
    ) {
    }

    //Getter
    public function __get(string $list):mixed
    {
        if (property_exists('ToDoList', $list)) {
            return $this->{$list};
        } else throw new Exception("Attribute " . $list . " does not exist
in class ToDoList!");
    }

    //Setter
    public function __set(string $entries, mixed $value):void
    {
        if (property_exists('ToDoList', $entries)) {
            $this->{$entries} = $value;
        } else throw new Exception("Attribute " . $entries . " does not
exist in class ToDoList!");
    }

    //Ein neuer Entry kann von jedem erstellt werden. Jedoch darf
    //der die Entry-ID noch nicht existieren.
    public function addEntry(ToDoListItem $entry):void
    {
        if (!array_key_exists($entry->entryId, $this->entries))
            $this->entries[$entry->entryId] = $entry;
        else throw new Exception("Entry " . $entry->entryId . " does not
exist!");
    }

    //Es ist nur für den Ersteller möglich, seine Einträge zu löschen.
    //Dieser werden hier aus dem Array mittels der Entry-ID gelöscht.
    public function deleteEntry(ToDoListItem $entry):void
    {
        if($this->loginUser->role == 1 ||
            ($this->loginUser->userId == $entry->creatorId &&
                array_key_exists($entry->entryId, $this->entries)))
            unset($this->entries[$entry->entryId]);
        else throw new Exception("Entry " . $entry->entryId . " cant be
deleted!");
    }
}
```



```

//Es ist hier wieder nur für den Ersteller möglich, seine
//Einträge zu bearbeiten. Er kann hier den Titel sowie
//den Text seines Eintrages ändern. Zudem ändert sich
//dann auch die Editor-ID und das Edit-Datum auf das
//heutige Datum.
public function editEntry(TodoListItem $entry, string $title, string
$text):void
{
    if($this->loginUser->role == 1 ||
        $this->loginUser->userId === $entry->creatorId &&
        array_key_exists($entry->entryId, $this->entries)) {
        $obj = $this->entries[$entry->entryId];
        $obj->title = $title;
        $obj->text = $text;
        $obj->editorId = $this->loginUser->userId;
        $obj->editDate = new DateTime();
    }
    else throw new Exception("Entry " . $entry->entryId . " cant be
edited!");
}

```

```

//Es ist hier wieder nur für den Ersteller möglich, seine
//Einträge abzuschließen. Hier ändert sich der Status.
public function finishEntry(TodoListItem $entry):void
{
    if($this->loginUser->role == 1 ||
        $this->loginUser->userId === $entry->creatorId &&
        array_key_exists($entry->entryId, $this->entries))
        $entry->status = "abgehakt";
    else throw new Exception("Entry " . $entry->entryId . " cant be
finished!");
}

```

```

//Magic Method __toString() für die Ausgabe der
//ToDoListe für den gerade angemeldeten User
public function __toString():string
{
    $result="<div class='welcome'>\n
<h1>DEINE TO-DO LISTE</h1>\n
<h2>Hallo ".$this->loginUser->userId."</h2>\n
</div>\n";
    $result.="<div class='ToDoList'>\n";

```

```

        foreach($this->entries as $entry) {
            //Zeigt nur Entries vom angemeldeten User, außer bei Admin
            if ($entry->creatorId === $this->loginUser->userId || $this-
>loginUser->role == 1)
        ) {
            $result .= $entry;
            $result .= "<div class='buttons'>\n<input type='submit'
class='edit' value='Bearbeiten'>";
            $result .= "<input type='submit' class='delete'
value='Löschen'>";
            $result .= "<input type='submit' class='finish'
value='Abschließen'>\n</div>";
            $result .= "\n</div>";
        }
    }

    $result .= "\n</div>";
    return $result;
}

```

TodoListItem.php

```
<?php

class TodoListItem
{
    public function __construct(
        private string $entryId,
        private DateTime $creationDate,
        private string $creatorId,
        private string $title,
        private DateTime $editDate,
        private string $text = "",
        private string $status = "aktiv",
        private string $editorId = "",
    )
    {
    }

    //Getter
    public function __get(string $entry):mixed
    {
        if (property_exists('TodoListItem', $entry)) {
            return $this->{$entry};
        } else throw new Exception("Attribute " . $entry . " does not exist
in class TodoListItem!");
    }

    //Setter
    public function __set(string $entry, mixed $mValue):void
    {
        if (property_exists('TodoListItem', $entry)) {
            $this->{$entry} = $mValue;
        } else throw new Exception("Attribute " . $entry . " does not exist
in class TodoListItem!");
    }

    //Magic Method __toString() für die Ausgabe der
    //einzelnen Einträge in der TodoListe
    public function __toString():string
    {
        $result = "\n<div id='".$this->entryId.'" class='entry'>
            <div class='status'>Status: ".$this->status."</div>
            <h3 class='title'>Titel: ".$this->title."</h3>";

        if($this->text != "")
            $result .= "<p class='note'>".$this->text."</p>";

        $result .= "<p>Ersteller: ".$this->creatorId."</p>
            <p>Erstellungsdatum: ".$this->creationDate-
>format("d.m.Y")."</p>";

        //Erst wenn eine Bearbeitung stattfand soll auch das
        // Bearbeitungsdatum und die ID des Bearbeiters angezeigt werden
        if($this->editorId != "") {
            $result .= "<p>Letzter Bearbeiter: " . $this->editorId .
"</p>";
            $result .= "<p>Letztes Bearbeitungsdatum: " . $this->editDate-
>format("d.m.Y") . "</p>";
        }
        return $result;
    }
}
```

```
}  
}
```

UserList.php

```
<?php
```

```
class UserList{  
    public function __construct(  
        private array $users = []  
    )  
    {  
    }  
  
    public function __get(string $users): mixed  
    {  
        if (property_exists('UserList', $users)) {  
            return $this->{$users};  
        } else throw new Exception("Attribute " . $users . " does not exist  
in class User!");  
    }  
  
    public function __set(string $users, mixed $newValue): void  
    {  
        if (property_exists('UserList', $users)) {  
            $this->{$users};  
        } else throw new Exception("Attribute " . $users . " does not exist  
in class User!");  
    }  
  
    public function addUser(User $user){  
        if (!array_key_exists($user->userId, $this->users)){  
            $this->users[$user->userId] = $user;  
        }  
        else throw new Exception("User " . $user->userId . " already  
exists!</div>");  
    }  
}
```

User.php

```
<?php
```

```
class User
{
    public function __construct(
        private string $userId,
        private string $password,
        private string $email,
        private int $role = 0,
        private array $entries = []
    )
    {
    }

    public function __get(string $user): mixed
    {
        if (property_exists('User', $user)) {
            return $this->{$user};
        } else throw new Exception("Attribute " . $user . " does not exist
in class User!");
    }

    public function __set(string $user, mixed $newValue): void
    {
        if (property_exists('User', $user)) {
            $this->{$user};
        } else throw new Exception("Attribute " . $user . " does not exist
in class User!");
    }
}
```

style.css

```
body{
  position: relative;
  font-family: "Bahnschrift", sans-serif;
  box-sizing: border-box;
  background: url("../img/bg.jpg");
}

.welcome{
  width: 100%;
  background-color: #fff;
  text-align: center;
  color: #5e8949;
  padding: 10px;
  border-bottom: 8px solid #5e8949;
}

h1, h2{
  margin: 0;
  font-size: 28px !important;
}

.title{
  text-align: left;
  font-size: 18px !important;
}

.login_form, .registration_form, .notify{
  position: absolute;
  padding: 20px 50px 5px 50px;
  background-color: white;
  left: 35vw;
  width: 30%;
}

.login_form{
  top: 13vh;
}

.registration_form{
  top: 43vh;
  z-index: -1;
}

.registration_form h3{
  z-index: 999;
}

i{
  color: red;
}

.notify{
  margin-top: 40px;
  font-size: 20px;
  text-align: center;
}
```

```

.red{
  color: red;
}

.green{
  color: green;
}

form input{
  box-sizing: border-box;
  border-radius: 5px;
  border: 1px solid #999;
  padding: 4px;
  outline: none;
}

.reg_button, .login_button, .logout_button{
  font-size: 1.1em;
  padding: 10px;
  font-family: Roboto, sans-serif;
  font-weight: 300;
  color: #fff;
  background-color: #5e8949;
  border-radius: 20px;
  cursor: pointer;
  margin-bottom: 2em;
  margin-top: 15px;
}

.logout_button{
  position: absolute;
  top: 10px;
  right: 30px;
}

.toDoList{
  width: 100%;
  padding: 7%;
  display: flex;
  gap: 50px;
  flex-wrap: wrap;
}

.entry{
  display: flex;
  justify-content: space-evenly;
  flex-direction: column;
  background-color: #fff;
  color: #5e8949;
  padding: 30px;
  border-radius: 20px;
  font-size: 13px;
  width: 400px;
}

.entry h3{
  margin-bottom: 20px;
}

```

```

.status{
  background-color: #aaa;
  color: #fff;
  padding: 20px;
  text-transform: uppercase;
  margin-bottom: 20px;
  font-size: 15px;
  text-align: center;
}

.note{
  color: #aaa;
  font-size: 14px;
  margin-bottom: 20px;
}

.buttons{
  display: flex;
  justify-content: center;
  gap: 20px;
  text-align: center;
}

.delete, .edit, .finish{
  background-color: #aaa;
  color: #fff;
  padding: 10px;
  border-radius: 10px;
  cursor: pointer;
  font-size: 15px;
  border: none;
  width: 30%;
}

label{
  display: inline-block;
  width: 100px;
  margin: 2px;
}

input{
  display: inline-block;
  margin: 2px;
}

.error{
  border-color: red;
}

.form-style-5{
  max-width: 500px;
  background: #f4f7f8;
  margin: 10px auto;
  padding: 20px;
  border-radius: 8px;
}

.form-style-5 fieldset{
  border: none;
}

```

```

.form-style-5 legend {
    font-size: 1.4em;
    margin-bottom: 10px;
}

.form-style-5 label {
    display: block;
    margin-bottom: 8px;
}

.form-style-5 input[type="text"],
.form-style-5 textarea{
    background: rgba(255,255,255,.1);
    border: none;
    border-radius: 4px;
    font-size: 15px;
    outline: 0;
    padding: 10px;
    width: 100%;
    box-sizing: border-box;
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    background-color: #e8eeef;
    color: #8a97a0;
    -webkit-box-shadow: 0 1px 0 rgba(0,0,0,0.03) inset;
    box-shadow: 0 1px 0 rgba(0,0,0,0.03) inset;
    margin-top: 20px;
}

.form-style-5 input[type="text"]:focus,
.form-style-5 textarea:focus{
    background: #d2d9dd;
}

.form-style-5 select{
    height: 35px;
}

.form-style-5 .number {
    background: #5e8949;
    color: #fff;
    height: 30px;
    width: 30px;
    display: inline-block;
    font-size: 0.8em;
    margin-right: 4px;
    line-height: 30px;
    text-align: center;
    text-shadow: 0 1px 0 rgba(255,255,255,0.2);
    border-radius: 15px 15px 15px 0px;
}

```



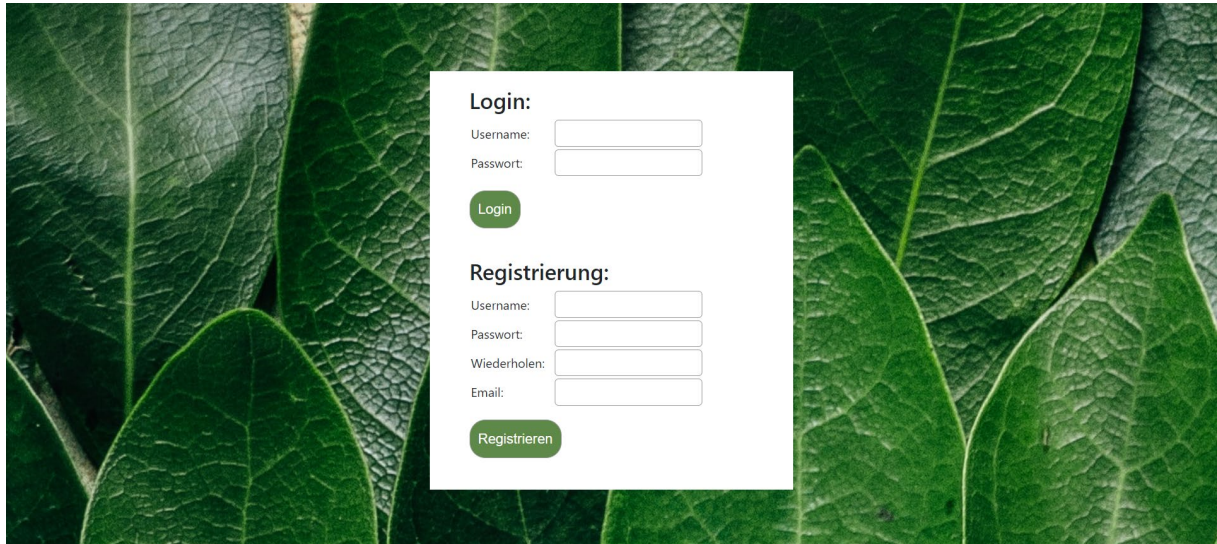
```

.form-style-5 input[type="submit"],
.form-style-5 input[type="button"]
{
    position: relative;
    display: block;
    padding: 19px 39px 18px 39px;
    color: #FFF;
    margin: 0 auto;
    background: #5e8949;
    font-size: 18px;
    text-align: center;
    font-style: normal;
    width: 100%;
    border: 1px solid #5e8949;
    border-width: 1px 1px 3px;
    margin-bottom: 10px;
}
input[type="submit"]:hover
{
    background: #6eaa69;
    transition: 0.5s ease;
}

```

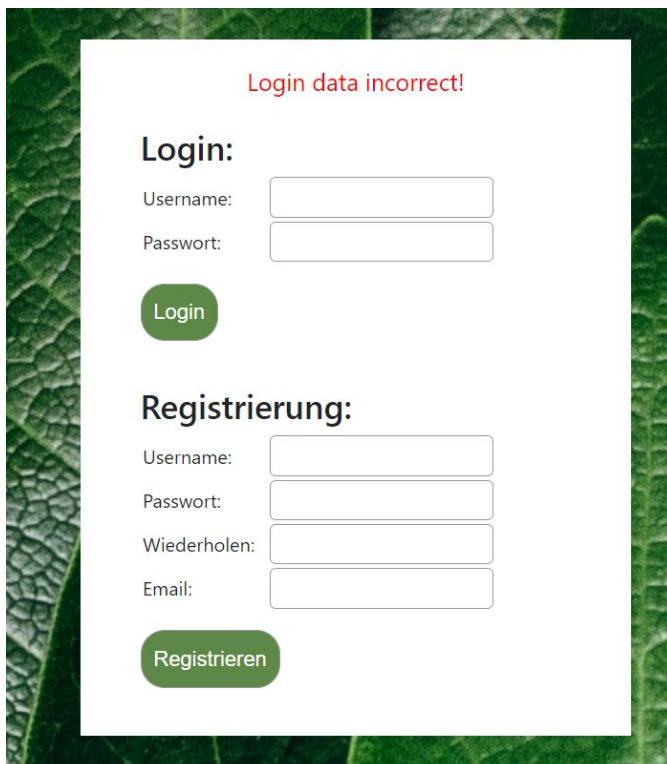
Testfälle

Login- und Registrierungsseite



The screenshot shows a web form for login and registration. The form is white and centered on a background of green leaves. It has two main sections: 'Login:' and 'Registrierung:'. The 'Login:' section has two input fields for 'Username:' and 'Passwort:', followed by a green 'Login' button. The 'Registrierung:' section has four input fields for 'Username:', 'Passwort:', 'Wiederholen:', and 'Email:', followed by a green 'Registrieren' button.

Fehler: keine Logindaten eingegeben



The screenshot shows the same web form as before, but with an error message. At the top of the form, the text 'Login data incorrect!' is displayed in red. The 'Login:' section has two empty input fields for 'Username:' and 'Passwort:', followed by a green 'Login' button. The 'Registrierung:' section has four empty input fields for 'Username:', 'Passwort:', 'Wiederholen:', and 'Email:', followed by a green 'Registrieren' button.

Fehler: keine/falsche Registrierungsdaten eingegeben

Registration data not correct!

Login:

Username:

Passwort:

Login

Registrierung:

Username:

Username must be at least 6 characters long

Passwort:

Password must be at least 8 characters long and must match

Wiederholen:

Email:

Must be a valid email address

Registrieren

Auch Eingaben bleiben erhalten

Registration data not correct!

Login:

Username:

Passwort:

Login

Registrierung:

Username:

Passwort:

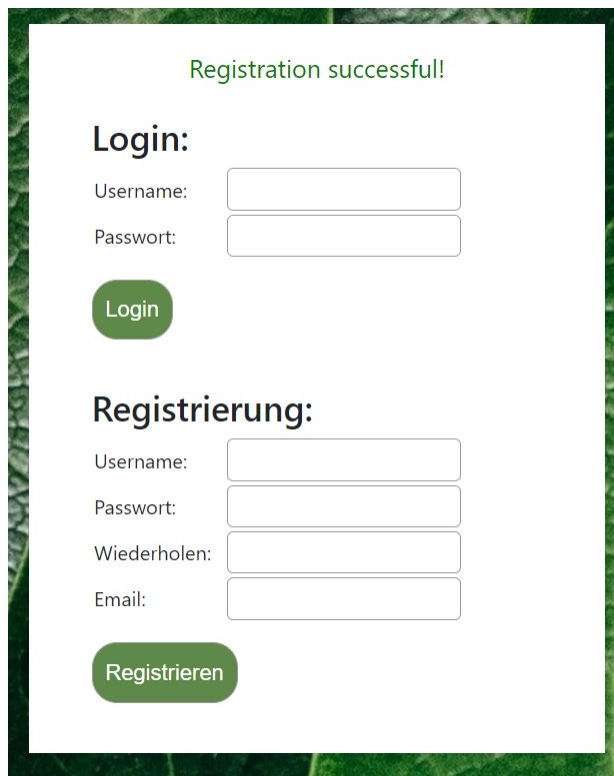
Password must be at least 8 characters long and must match

Wiederholen:

Email:

Registrieren

Erfolgreiche Registrierung



Registration successful!

Login:

Username:

Passwort:

Login

Registrierung:

Username:

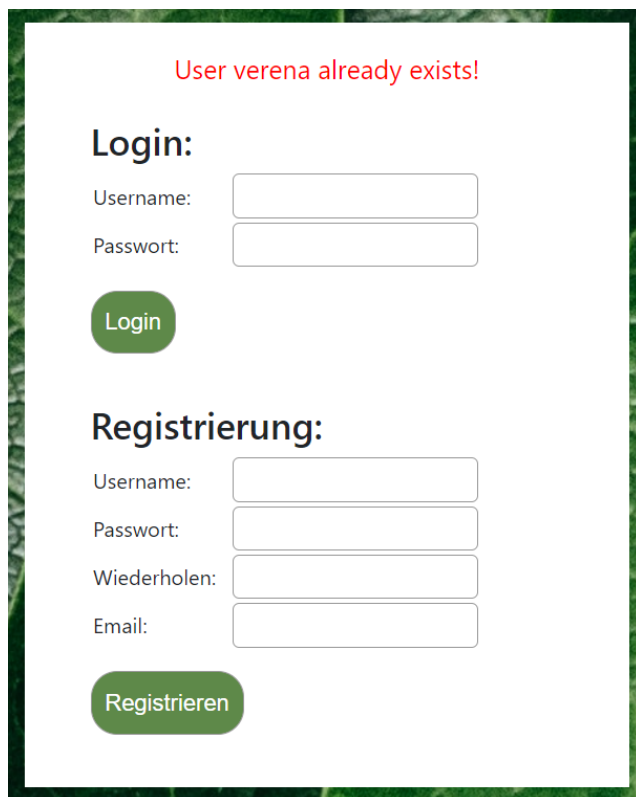
Passwort:

Wiederholen:

Email:

Registrieren

Fehler: Registrierung -> User existiert bereits



User verena already exists!

Login:

Username:

Passwort:

Login

Registrierung:

Username:

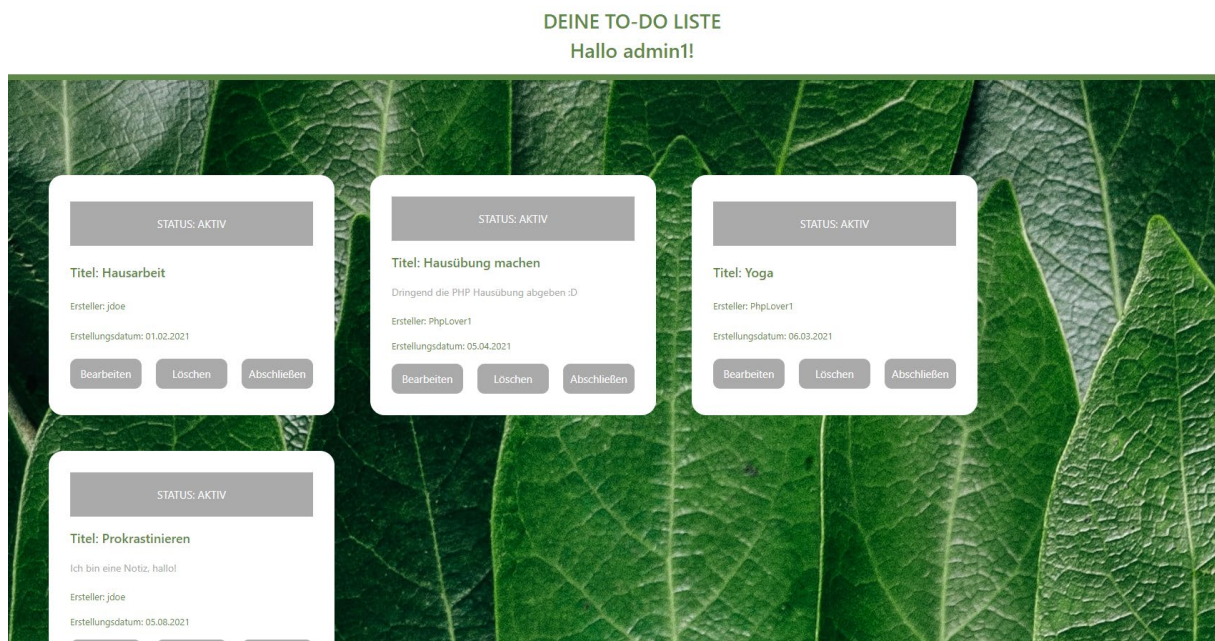
Passwort:

Wiederholen:

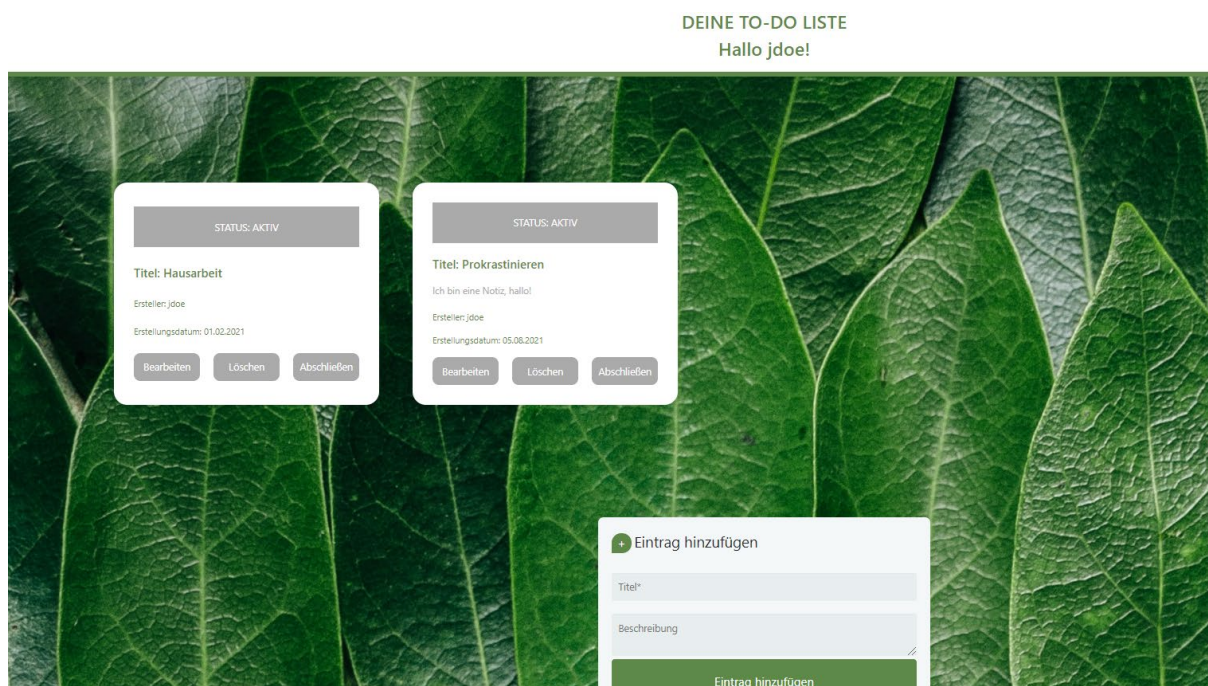
Email:

Registrieren

Admin eingeloggt -> sieht alle Tasks



Liste von User jdoe mit reingeladenen Daten -> nur seine Tasks sichtbar



Fehler: Eintrag hinzufügen (nur Titel verpflichtend, darum keine Fehler bei Description)

+ Eintrag hinzufügen

Abc

Title must be at least 6 characters long!

Eintrag hinzufügen

Neuer Eintrag hinzugefügt

