Projekt uge 8 - To do

Gruppemedlemmer: Sophie, Mikkel, Tobias.
https://github.com/toby044/toDoProject

Indholdsfortegnelse

Indledning

I vores gruppeprojekt skal vi udvikle en “to do” applikation, som skal udvikles med henblik på opbevaring af data i en database. I vores opgave vil vi lave et login-system, som skal kunne verificere en bruger via sammenligning med databasen og logge ind, samt få vist egne unikke todos. Vi vil udvikle vores applikation med backend i PHP og MySQL, per definition af projektbeskrivelsen.

Resume

Til dette gruppeprojekt er der blevet udarbejdet et website som løsning til problemformuleringen. Vi har skabt en “to do” applikation, som tillader individuelle brugere at oprette “todos”.

Vi har udviklet denne applikation til succes med et design skabt i Figma, videreført i PHP, HTML, CSS og MySQL.

Disse todos kan enten være aktive eller checked. To do siden (index hvis man er logget ind) viser alle aktive todos, og historik siden viser alle man har på sin bruger, fuldført eller ej.

Når en bruger oprettes, gemmer vi dataen i vores database. Denne data indeholder et unikt id, brugernavn og et hashed password, som vi har skabt med bcrypt funktionen. Det unikke id trækker todos fra databasen, som matcher en foreign key, og på to do siden genererer vi det rigtige indhold til den individuelle bruger.

Når en bruger skal logge ind, checker vi også at det er den rigtige bruger ved at checke brugernavnet og password med funktionen password_verify. På denne måde kan vi ikke vide hvilket password brugeren har, som skaber sikkerhed for brugeren.

Der er tilsvarende en admin side, hvor en bruger med rollen “admin” kan slette brugere.

Problemformulering

“Hvordan udvikler vi bedst muligt en “to do” applikation, som tillader oprettelse af beskrivende todos med startdato, samt mulighed for arkivering og bruger login system?”

Problemstilling

Projektet lyder på at udvikle en todo applikation, hvor man, som bruger, har mulighed for at registrere sig, oprette og “tjekke” private todos af. Værdierne skal gemmes i en database hvori en admin også har mulighed for at redigere i brugerlisten. Dertil skal der også være en funktionalitet der outputter en brugers todos til et JSON format.

Metodeovervejelser

Trello

Vi vil bruge Trello til opgave overblik, samt tidsplan. Her kan alle gruppemedlemmer nemt danne sig overblik over, hvad der mangler og ikke mangler at blive udviklet.

Figma

Vi vil bruge Figma til idegenerering af app designet, samt opstilling af ER-diagram for overblik over databasestrukturen.

PHP & MySQL

Vi vil bruge PHP i forbindelse med MySQL til udviklingen af projektet. Vi har arbejdet med MySQL i forlængelse af phpMyAdmin.

GitHub

Vi vil bruge GitHub til et remote repository, som gruppemedlemmerne kan tilgå for at redigere og opdatere projektet.

Research & Analyse

Til projektet har vi umiddelbart brugt, hvad vi har lært indtil videre fra pensum og så vidt muligt holdt os inde for det. Vi har dog researchet os frem til andre metoder, men fravalgt at bruge dem og forsøgt at bruge det vi har lært til at opnå samme resultat. Dertil skulle vi finde ud af hvordan man bruger bcrypt kryptering til brugernes kodeord, samt lave en funktion der konvertere brugernes todos til JSON format.

JSON

For at kunne outputte en brugers todos har vi gjort følgende. Her tager vi fat i den cookie som bliver oprettet ved brugerlogin og vælger alt i den brugers “todos”. Herefter iterere vi gennem det resultat og placere det i et PHP array. Så kan man udnytte det på en af to måder:

- Printe indholdet på siden (linje 17)
- Konvertere dataen til en JSON fil format (linje 20-22)

Til sidst lukker vi forbindelsen til databasen.

```php
<?php
// Database forbindelse
include 'db_conn.php';

// Vælg brugers todos og opbevar i en variabel
$userid = $_COOKIE['userid'];
$sql = "SELECT * FROM todos WHERE user_id = '$userid'";
$result = mysqli_query($conn, $sql);

// Konverter MySQL til PHP array og loop igennem det
$arr = array();
while($row =mysqli_fetch_assoc($result)){
  $arr[] = $row;
}

// Printer JSON String format
echo json_encode($arr);

// Konvertere til JSON fil format
$fp = fopen('userdata.json', 'w');
fwrite($fp, json_encode($arr));
fclose($fp);

// Luk database forbindelse
mysqli_close($conn);
```

Bcrypt

I vores opgave hasher vi passwords med phps indbyggede bcrypt funktion. Vi anvender den ved at tage værdierne fra vores post og hasher ind i en ny variabel på linje 28, som vi til sidst skubber ind i databasen med name og password. Når der bliver tilføjet en bruger til databasen, får brugeren også automatisk et ID som bliver auto-inkrementeret, samt en rolle som er 'user' by default.

```php
        $username = mysqli_real_escape_string($conn, $_POST['username']);
        $password = mysqli_real_escape_string($conn, $_POST['password']);

        $pass = password_hash($password, PASSWORD_BCRYPT);

        // lav sql
        $sql = "INSERT INTO users(name, password) VALUES('$username','$pass')";
```

Sådan kan et hashet password se ud:

password
$2y$10$sNTXw3U49uAfvmT3qJJs6OQkYl.AkN4wHl8XPhKC7H4...

Konstruktion

Wireframe & ER-Diagram

[Link til wireframe]

Som starten på konstruktionsfasen har vi designet et udkast til, hvordan vi ville have applikationen til at se ud, samt vores database struktur via et ER-Diagram.

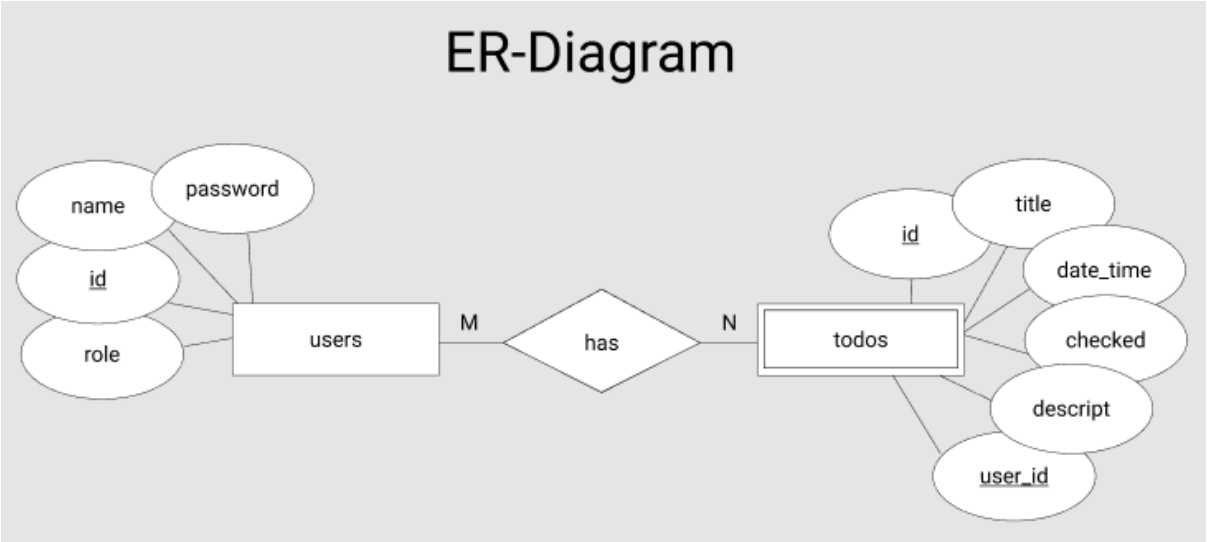
ER-Diagrammet tager udgangspunkt i entiteten ***users*** som har attributterne:

- ***id*** (primary key)
- ***name*** (det valgte brugernavn ved registrering)
- ***password*** (det valgte kodeord ved registrering)
- ***role*** (om en bruger er ***user*** eller ***admin***)

Derefter har vi et mange til mange forhold til en svag entitet ***todos*** som har attributterne:

- ***id*** (primary key)
- ***title*** (titlen af en todo)
- ***date_time*** (tidspunktet man opretter en todo på)
- ***checked*** (boolean værdi: om den er tjekket af eller ej)
- ***descript*** (uddybning af en todo, hvis ønsket)
- ***user_id*** (foreign key)

De fleste attributter printes ud på siden via ***echo***.
En todos ***user_id*** bliver forbundet med brugerens ***id***, så man kun ser de todos der høre til den givne bruger.



Database

Vi vælger at bruge phpmyadmin, til at lave vores database. Vi vælger dette pga brugervenligheden der gør det let og ligetil at arbejde med, da der er et interface som vi

bruge til opdatering af forskellige sql-sætninger og ændringer i brugere. Som set i vores ER-Diagram, vælger vi at have to forskellige tables som vi forbinder med en foreign key.

users table:

Navn	Datatype	Tegnsæt (sortering)	Attributter	Nulværdi	Standardværdi	Kommentarer	Ekstra	Handling
id	int(11)			Nej	*Ingen*		AUTO_INCREMENT	Ret Slet Mere
name	varchar(50)	utf8mb4_general_ci		Nej	*Ingen*			Ret Slet Mere
password	varchar(200)	utf8mb4_general_ci		Nej	*Ingen*			Ret Slet Mere
role	varchar(50)	utf8mb4_general_ci		Nej	user			Ret Slet Mere

Vores brugere har et id, name, password som bliver hashet og en rolle, som determinerer hvilken side der bliver frembragt, når man går til index siden. Rollen har en standard værdi som 'user' sådan at nye brugere ikke får rollen af admin.

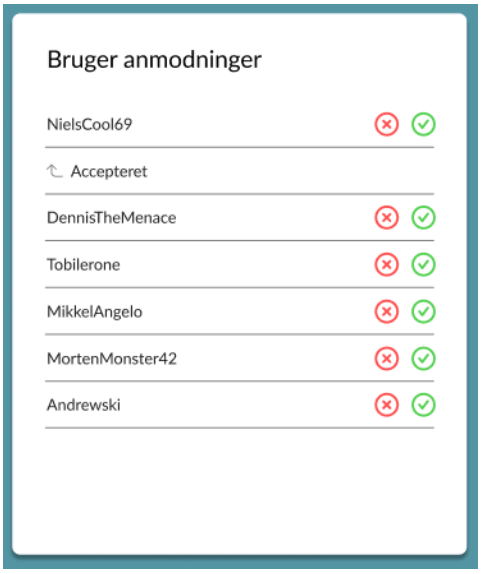
ID er vores primary key, som vi bruger til at få forbindelse til vores todos table.

todos table

Navn	Datatype	Tegnsæt (sortering)	Attributter	Nulværdi	Standardværdi
id	int(11)			Nej	*Ingen*
title	text	utf8mb4_general_ci		Nej	*Ingen*
descript	text	utf8mb4_general_ci		Nej	*Ingen*
date_time	datetime			Nej	current_timestamp()
checked	tinyint(1)			Nej	0
user_id	int(11)			Nej	*Ingen*

Vores todos indeholder alle todos som brugerne har. Den enkelte todo har en id, titel, descript, date_time, checked og user_id. **id** bliver brugt til at give den individuelle todo et unikt id, som vi kan hente data fra. **title** bliver brugt til at give en titel vores todos, samme med **descript**, som giver en beskrivelse til det. **date_time** sætter tiden både oprettet og deadline. **checked** bruger vi til at bestemme hvilke der skal vises som aktive todos, og hvilke der kun skal lægge i historikken. **user_id** bruger vi som fremmednøgle, og her trækker vi alle de todos ud, som den bestemte bruger har. Dette gør vi ved hjælp af cookies i browseren.

Administrationsdel



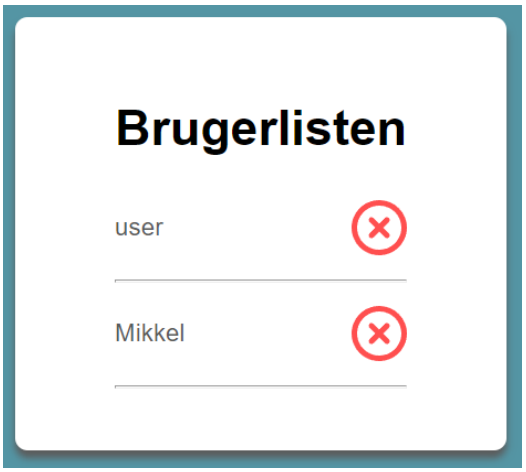
Vores admin design startede vi ud med at se sådan ud, hvor man både kunne godkende og slette nye brugere.

Det som vi har endt med at lave er, at man ikke kan godkende brugere da de er godkendt i forvejen, så man ser alle brugere der bruger applikationen, og dem kan man så slette.

Fordi vi har gjort det sådan er, at vi synes det gav bedre mening til vores applikation, da vi ikke synes at brugeren skal vente på at blive godkendt for at bruge applikationen.

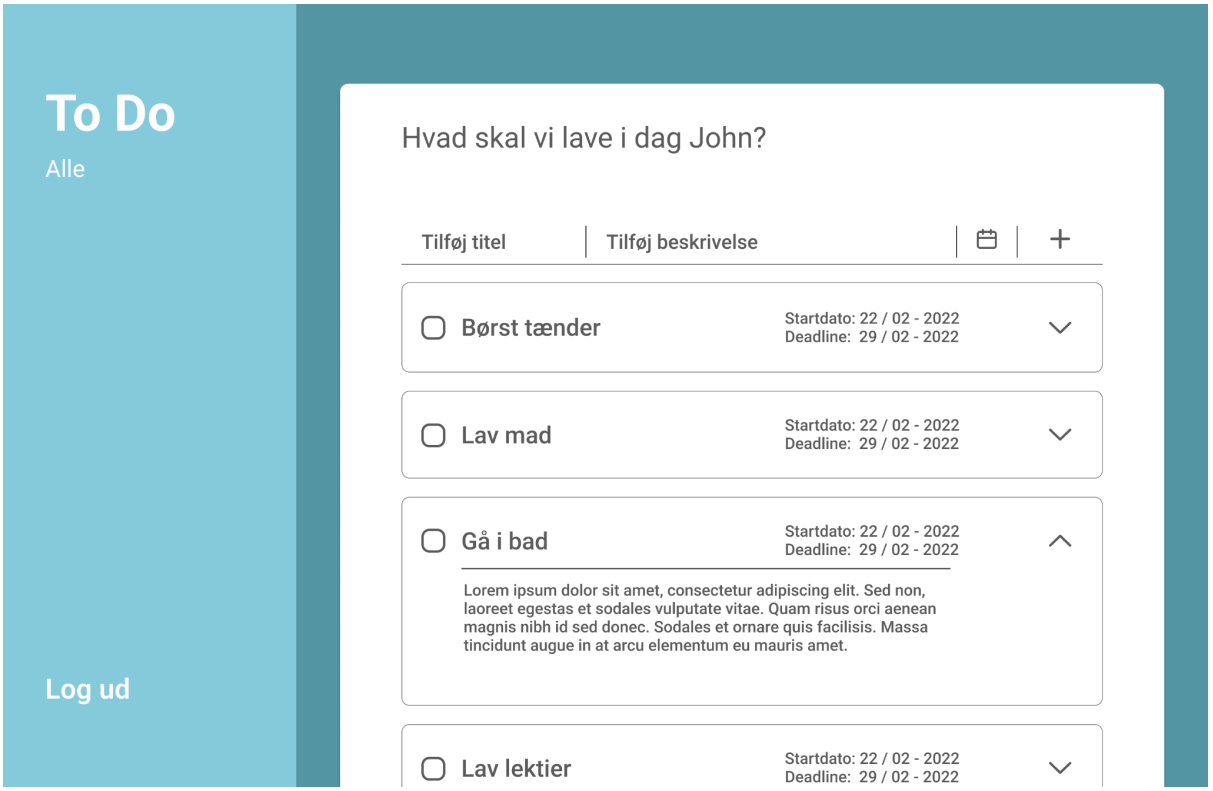
For at lave admin-siden skal man inkludere databasen, så man kan liste alle brugerne på siden, samt når administratoren trykker på en knap for at slette en bruger, så skal brugeren slettes fra databasen. I PHP inkluderer man databasen ved at skrive: **include('db_conn.php');**. For at liste brugerne ud på siden har vi skrevet et foreach loop i PHP, og inde i det loop er der et p tag der indeholder **<?php echo htmlspecialchars($databyte['name']);?>**. På den måde får man alle brugerne ud på en liste. Herfra har vi lavet en knap/input til hver brugere, som gør at den sletter brugeren fra databasen. For at forbinde inputet til databasen har vi først skrevet det value fra inputets name attribute i et if stament som ser sådan ud: **if(isset($_POST['notAccepted']))**. Inde i if stamentet skal der så stå **$select = "DELETE FROM users WHERE id = '$id'";**, som er det der gør at brugeren bliver slettet fra databasen.

Resultatet af siden ser sådan ud:



Brugerdel

Brugerne skal have muligheden for først at kunne registrere sig i systemet og derefter logge ind på applikationen. Her bliver brugerne mødt med en form hvori de kan oprette en todo med en uddybende beskrivelse, hvis ønsket. På submit bliver todo'en tilføjet til databasen og bliver vist på skærmen lidt under med en startdato og en deadline (som er fast på en uge, men vi kommer på dette igen i videreudvikling). Hvis ønsket kan brugeren udvide hver todo for at læse den udvidede beskrivelse via pilen til højre og når de er færdig med opgaven kan de tjekke den af og den vil blive flyttet til historik, hvor alle de udførte todos vil ligge.



Inden vi tilføjer og printer værdierne fra databasen ud, tjekker vi om brugeren har udfyldt de krævede felter og ellers beder vi brugeren indtaste en titel værdi som minimum.

```
<div class="todo-add-section">
  <form action="add-todo.php" method="POST" autocomplete="off">
  <?php if(isset($_GET['mess']) && $_GET['mess'] == 'error') { ?>
    <input type="hidden"
           name="user_id">
    <input id="todo-input-text"
                type="text"
                name="title"
                class="error"
                placeholder="Feltet må ikke være tomt" />
    <input id="todo-input-description"
                type="text"
                name="descript"
                placeholder="Beskrivelsen må ikke være tom" />
    <button type="submit" name="submit"><i class="fa fa-solid fa-plus font-size"></i></button>
  <?php } else { ?>
    <input id="todo-input-text"
                type="text"
                name="title"
                placeholder="Tilføj titel" />
    <input id="todo-input-description"
                type="text"
                name="descript"
                placeholder="Tilføj beskrivelse" />
    <button type="submit" name="submit"><i class="fa fa-solid fa-plus font-size"></i></button>
  <?php } ?>
  </form>
</div>
```

Hvis dette er sandt printer vi den todo de lige har tilføjet. En todo printes ud ved at loope gennem de værdier vi får fra databasen og via **echo** vises værdierne i HTML'en eller bruges til at tage fat i den enkelte todo via dets ***id***.

```php
<div class="todos-section">
  <?php foreach($result as $todo) { ?>
    <?php if (!$todo['checked']) { ?>
      <div class="todo-item b-radius" id="<?php echo $todo['id']; ?>">
        <div class="wrapper">
          <div class="wrapper-inner">
            <label class="todo-checkbox-container">
              <form action="index.php" method="POST">
                <input type="hidden" name="id" value="<?php echo $todo['id'];?>">
                <input type="submit" name="check">
              </form>
              <span class="todo-checkmark b-radius"></span>
            </label>
            <h2 class="todo-title"><?php echo $todo['title']; ?></h2>
            <small class= "todo-date">
              <?php
                $timestamp = $todo['date_time'];
                $expireDate = strtotime('+7days', strtotime($timestamp));
                echo 'Oprettet: ' . date('d-m-Y H:i:s', strtotime($timestamp)) . '<br/>' . 'Deadline: ' . date('d-m-Y H:i:s', $expireDate);
              ?>
            </small>
            <div class="dropdown-arrow">
              <a href="#<?php echo $todo['id']; ?>"><i class="fa fa-solid fa-chevron-down font-size"></i></a>
            </div>
          </div>
          <div class="wrapper-inner-2">
            <p class="todo-descript"><?php echo $todo['descript']; ?></p>
            <div class="todo-archive">
              <i class="fa fa-solid fa-box-archive"></i>
            </div>
          </div>
        </div>
      </div>
    <?php } ?>
  <?php } ?>
</div>
```

Moduler

Header

```php
if (isset($_COOKIE['uname'])) {
    echo '<aside>
            <div class="flex-down">
                <h2><a href="index.php" class="header-link">To Do</a></h2>
                <a href="all.php">Historik</a>
            </div>
            <a href="modules/logout.php" class="logoutBtn">Log ud</a>
        </aside>';
} else {
    echo '<aside>
            <div class="flex-down">
                <h2>To Do</h2>
            </div>
        </aside>';
}
```

Vores header er et aside element, som vi generer i en if-sætning. Denne sætning checker om man har en cookie, og genererer efterfølgende headeren vi vil have på login hvis man ikke har den cookie, og ellers det vi vil vise på to do siden.

Logout

```php
<?php
    setcookie('uname', null, -1, '/');
    setcookie('userid' , null , -1 , '/');

    header('Location: ../');

```

Her refererer vi til i vores header, når der skal logges en bruger ud. Vi fjerner bare en cookie og henviser til index siden, som generer login siden automatisk.

Login-system

Vi har i vores projekt brugt cookies som grundstenene til vores login. Når man kommer ind på vores index side, bestemmer vi nemlig, hvad man skal se. Da vi bruger forskellige php filer som moduler, kan vi nemt inkludere det content vi gerne vil have vist. For at checke om man er logget ind tidligere, kigger vi om der eksisterer en cookie med navnet **uname**, og hvis der ikke er en cookie, skal man logge ind. Har man en cookie, bliver der naturligt genereret en side hvor alle brugerens todos bliver vist.

```php
if(isset($_COOKIE['uname'])){
  include './modules/toDo.php';
} else {
  include './login/login.php';
}
```

Oprettelse af bruger

```php
if (!empty($_POST['register'])) {
    // Check brugernavn
    if (empty($_POST['username'])) {
        echo 'Indtast venligst brugernavn';
    } else {
        $username = $_POST['username'];
    }

    // Check password
    if (empty($_POST['password'])) {
        echo 'Indtast venligst password';
    } else {
        $password = $_POST['password'];
    }

    // Slut POST check

    // lav variabler
    $username = mysqli_real_escape_string($conn, $_POST['username']);
    $password = mysqli_real_escape_string($conn, $_POST['password']);

    $pass = password_hash($password, PASSWORD_BCRYPT);

    // lav sql
    $sql = "INSERT INTO users(name, password) VALUES('$username','$pass')";

    // Gem i databasen
    if (mysqli_query($conn, $sql)) {
        // Ved succes gå til index med logged in detaljer
        header('Location: ../index.php');
    } else {
        echo 'query error: ' . mysqli_error($conn);
    }
}
```

For at registrere en bruger, anvender vi mysql sætninger i forlængelse af en html form, hvor vi bruger POST objektet til at få værdier ud, som vi bruger med php. For at validere inputs fra brugeren, checker vi først om de har sat password og username ind på linje 9 og 16.

Har POST fået disse værdier, gemmer vi de indtastede værdier i variabler, som vi vil bruge til mysql. På linje 28 hasher vi vores password med bcrypt, sådan at man ikke kan se passworded i databasen, og på 31 sætter vi username og password værdierne ind i vores database, hvor brugeren får et unikt id og en rolle som bruger.

Efter indsætningen med sql checker vi om vores DB har modtaget det, og sender os derefter videre til index hvor vores todo content vil blive genereret.

Login med bruger

```php
// Hvis du er logget ind, gå til index
if (isset($_COOKIE['uname'])) {
    header("Location: ../index.php");
}

if (!empty($_POST['submit'])) {

    // Opret forbindelse til databasen

    // Set variables for post username and password
    $username = $_POST['username'];
    $password = $_POST['password'];
    $sql = "SELECT * FROM users WHERE name ='$username'";

    // Lav query med sql sætningen
    $qry = mysqli_query($conn, $sql);
    $arr = $qry->fetch_array(MYSQLI_ASSOC);
    $userid = $arr['id'];

    $count = mysqli_num_rows($qry);
    // Sætter cookie, som vi vil læse på og genere content med sql
    if ($count == 1) {
        if (password_verify($password, $arr['password']) === true) {
            if ($username == 'admin') {
                header("Location: ./admin.php");
            } else {
                header("Location: ./");
                setcookie('uname', $username, time() + 60 * 60 * 24 * 30, '/');
                setcookie('userid', $userid, time() + 60 * 60 * 24 * 30, '/');
            }
        }
    }
}
```

Når en bruger skal logge ind, skal den være oprettet. Er noget som vi gerne vil checke på samme måde som når man registrerer sig, altså med en html form der bruger POST metoden. De værdier som vi får ud af POST vil vi altså gerne checke på. Dette gør vi ved først at spørge databasen om vi overhovedet har en bruger som er derinde. Efter vi har gjort dette, trækker vi brugerens id ud, da vi gerne vil bruge den til at lave en cookie, som kan generere vores content på todo siden.

Da vores gemte passwords i databasen er hashed, bruger vi **password_verify();** på linje 27 til at checke passworded som bliver sat ind. Dette gør den ved at sammenligne det hashede password med det indskrevne, og matcher dem op imod hinanden.

Er passworded rigtigt, checker vi på linje 28 om brugeren hedder admin, og hvis han gør, bliver han viderestillet til admin siden, hvor han kan slette brugere.

Hvis det er en normal user, logger man i stedet ind, går til to do siden og får to cookies som varer 30 dage, som vi bruger til at display information og checke om man er logget ind, når vi vil åbne siden på et andet tidspunkt.

Videreudvikling

Grundet prioritering har vi ikke gjort det muligt for brugeren at kunne vælge en slutdato for en todo. Men dette kunne optimeres/tilføjes i fremtiden ved at tilføje et **date** input hvori brugeren kan vælge en dato, som bliver indført i databasen sammen med de andre værdier. Endnu en optimering hertil ville være at vise en dynamisk nedtælling til deadlinen på en todo.

Dertil har vi ikke gjort det muligt for brugeren at kunne prioritere deres todos efter start/slutdato, men dette kunne løses ved at tilføje en filterfunktion der gør det muligt at liste todos i stigende eller faldende orden.

Evaluering af proces

Arbejdsproces

Overordnet set har vores proces været tilfredsstillende. Da der som sådan ikke er fokus på design ved dette projekt, har vi været hurtigt over designfasen, så vi kunne fokusere på funktionaliteten. Vi har så vidt muligt forsøgt at dele konstruktionen af koden op, så vi alle er med til udvikle en del af applikationen. Her har vi også haft mulighed for at “ping ponge” med hinanden for at nå frem til et tilfredsstillende resultat sammen.

Produkt

Vores produkt er blevet stort set som vi havde set det for os da vi startede projektet. Vi har opnået projekt beskrivelsens must haves på tilfredsstillende vis, dog har vi valgt at ændre lidt af beskrivelsen da vi vurderede at det ikke var nødvendigt for en administrator at skulle til at acceptere alle nye brugere, da dette realistisk set ville resultere i at brugerne ville blive frustreret og ikke benytte app’en, da de ikke ville kunne logge ind umiddelbart efter registrering. Produktet har enkelte skønhedsfejl bl.a. at hvis brugeren ikke eksistere, så ved den ikke hvad den skal gøre og melder derfor en fejl på siden. Dette er ikke en dealbreaker men en fejl der skal rettes ved videreudvikling.

Konklusion

Vi har endt med et fuldt funktionelt produkt via PHP og MySQL, hvori brugerne kan registrere sig, logge ind og oprette todos, som kun er tilgængelige for den enkelte bruger og administratoren har muligheden for at se og fjerne brugere, efter ønske. Vi har her demonstreret forståelse for syntaks og brug af både PHP og MySQL til et fyldestgørende niveau, samt påvist evner inden for problemløsning.

Referencer

PHP og MySQL database
https://codewithawa.com/posts/to-do-list-application-using-php-and-mysql-database

Inspiration til animeret tekstbokse via CSS
http://jsfiddle.net/davidThomas/AY6Kt/

Log ind system (generel problemsøgning)
https://stackoverflow.com/

Todo
https://www.w3schools.com/sql/sql_orderby.asp
https://www.w3schools.com/php/php_mysql_insert.asp
https://www.studentstutorial.com/php/php-mysql-data-insert.php

Inspiration til todo checkbox funktion
https://www.youtube.com/watch?v=hKBBx9uqID4
https://itqna.net/questions/19378/update-table-field-just-checked-checkbox-php-mysql

MySQL til JSON via PHP
https://www.kodingmadesimple.com/2015/01/convert-mysql-to-json-using-php.html
https://www.youtube.com/watch?v=I4SRqAS7J8U

Administrationssiden
https://www.youtube.com/watch?v=TYkiR_21Y0M