Movie
Review

En platform hvor man kan danne sig et overblik af film, skuespillerer og læse sig frem til hvilke film der er gode

Jeg** har udviklet en **webapplikation** med lidt **JavaScript og PHP som hedder MovieReview. *Brugerne skal hurtig danne sig et overblik over film og skuespillere samt læse flere forksellige anmeldelser for, at afgøre, hvilke film der er værd at se.*

Sådan ser min database ud -en mange til mange

Jeg har 4 tabeller involveret & en connection tabel med følgende primary keys
De 4 tabeller:
Reviews - primary key: reviewId
Reviewers - primary key: reviewerId
Movies - primary key: movieId
Actors - primary key: actorId
Movie_actor - primary key: movieactorId

Connection tabellen:
Jeg har forbundet tabellerne ved at lave 4 kolonner der hver især er knyttet til reviewernes primary key. Jeg har altså *ikke* tilføjet en connection kolonne med et index i nogle af de 4 tabeller jeg bruger bare primary key'en.

INNER/LEFT JOIN movie_actor ON movieId = movieConId
INNER/LEFT JOIN movie_actor ON reviewerId= reviewerConId
INNER/LEFT JOIN movie_actor ON reviewId= reviewConId
INNER/LEFT JOIN movie_actor ON actorId= actorConId

Anmeldelser left join Reviews.php

Reviews.php afspiller alle anmeldelser og detaljer omkring anmeldelsen. Hvis man klikker på en anmeldelse skal man kunne læse den, hvis man klikker på anmelderen skal man kunne klikke på revieweren og se detaljer om personen.
Jeg henter mine anmeldelser ved en sql forespørgsel og lægger den ind i en variabel $reviews
Derefter laver jeg en *foreach* hvor jeg sørger for at hvert enkelt element afspilles nemt.
Ved at bruge foreach kan jeg få adgang til hver enkelt anmeldelse én ad gangen.
Ved at definerer *$review* i *foreach*-sætningen får jeg mulighed for at referere til den aktuelle anmeldelse inden for *loopet*. Dette betyder, at jeg nemmere kan få adgang til anmeldelsernes detaljer såsom som review_titel, review_text og afspille dem i min HTML.

Left join? Eller inner join?

I nogen af sql forespørgslerne anvender jeg *LEFT JOI*N for at sikre, at alle anmeldelser fra 'reviews' tabellen vises, selvom hvis der ikke er et match i de tilsluttede tabeller.
 Dette er vigtigt, fordi jeg ønsker at se alle anmeldelser, ikke kun dem der er tilknyttet andre tabeller. Hvis jeg havde brugt INNER JOIN, ville kun de anmeldelser, jeg havde connected til i de tilsluttede tabeller, blevet vist. Det vil gøre at de anmeldelser uden forbindelse til andre af de tilsuttede tabeller vil gå tabt.

3. Review.php denne side viser selve det review brugeren klikker på & afspiller skuespillerens film og filmens andre medvirkende samt

- Henter id'et og sender det som et parameter via url'en til min review.php side med $_GET
- Der er et array som holder styr på om filmen er vist

Tilføj anmeldelser post og loop-AddReviews.php

4. Addreview.php er en side hvor brugeren kan få lov til at skrive sin egen anmeldelse. Via $_post sendes brugerens/revieweren sin anmeldelse og både filmen+revieweren+reviewet connectes også i min connection tabel ved

1. laver en anonym funktion, men en addeventlistener til min stjerne(click)/stars.forEach er et loop som kører en funktion for hvert element i min liste (boostrapstjernere)
2. parametrene *star*,*index* (star er boostrapstjerner) index er "positionen af stjerne" (0, 1 eller 4)
3. i = 0, I<=INDEX; i kører op til index-af stjerner(antallet),
4.
5. FORM VALUE: rating sendes ind i input. og input valuen sendes via min betingelse if($_POST){}

Update sql & rediger din væg -

Skal det ikke kun være anmelderernes film der skal vises men også brugernes. Derudover skal der også være overblik over hvilke filminstruktører som har lavet hvad.

Fetch API og søgemaskine

Til projektet har jeg anvendt meget lidt javascript. Jeg har lavet en search.js og koblet den til min index side hvor jeg henter jeg film fra mit API med fetch() og har lavet en søgemaskine med film, som jeg har hentet fra API.
Gennemgår search.js og
Getmovies() (en funktion som henter film)
ShowMoves[en funktion som ved foreach afspiller hver film]
Til sidst er der tilføjet en event listener til min søgeformular. Søgeordet fra brugeren gemmes i konstanten *searchTerm* i inputfeltet. *En if-else-betingelse* kontrollerer, om brugeren har indtastet noget, og aktiverer *getMovies()* -funktionen, hvis det er tilfældet.

getMovies() denne funktion henter filmene.

Fetch(url,options) Jeg fetcher min url med de definerede options (som er objektet der indeholder autorisationen og headers) *.then(res⇒res.json)()* når data bliver modtaget håndteres de som en response ved at kalde .json(), får vi svaret som json-data *.then(data⇒{showMovies(data.results)}}* Her bliver data.results (listen med film) sendt til funktionen *showMovies()* . Funktionen *showMovies()* kan så bruge denne liste til at vise hver film på din side.

showMovies(data) denne funktion afspiller filmene.
mvoieContainer.innerHtml ='' tømmes
derefter laver jeg en forEach hvor movies er mit parameter. Min movies er et objekt med udvalgte informationer jeg hentede fra via fetch hentede fra mit data-array .Derefter laver jeg et div element og afspiller min informationer. Jeg sørger for at jeg skipper de film uden billeder Jeg tilføjer til sidst en mine film(div) til min html struktur (movieContainer).

searchForm.addEventListeners
Jeg har til min Form har jeg tilføjet en Addeventlistener.
Den "værdi" brugeren søgeord har jeg lagt ind i min konstant SearchTerm skriver inde i mit input felt.
Derefter laver jeg en if else med betingelsen, at hvis brugeren sørger
så skal getMovies funktion ske.

TALEPAPIR STARTER HERFRA

Movie
Review

En platform hvor man kan danne sig et overblik af film, skuespillerer og læse sig frem til hvilke film der er gode

Jeg *har udviklet en* ***webapplikation*** *med lidt* ***JavaScript og PHP*** som hedder MovieReview. *Brugerne skal hurtig danne sig et overblik over film og skuespillere samt læse flere forksellige anmeldelser for, at afgøre, hvilke film der er værd at se.*

●Introduktion dabatasestruktur

Reviews.php viser alle anmeldelser samt tilhørende detaljer. Ved at klikke på en anmeldelse kan man læse den, og ved at klikke på anmeldelsen kan man se detaljer om anmelders personen. Jeg henter anmeldelserne med en SQL-forespørgsel og gemmer dem i variablen $reviews. Derefter anvender jeg en foreach-løkke til at vise hver anmeldelse individuelt.

I nogle af SQL-forespørgslerne som i eksemplet med reviews.php anvender jeg LEFT JOIN for at sikre, at alle anmeldelser fra reviews-tabellen vises, selvom der ikke er et match i de tilsluttede tabeller.Det gør jeg fordi jeg vil se ALLE anmeldelser , ikke kun dem hvor der er et “match” i begge tabeller.

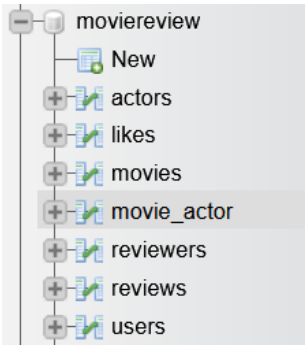
Sådan ser min database ud -en mange til mange

Jeg har 4 tabeller involveret & en connection tabel med følgende primary keys
De 4 tabeller:
Reviews - primary key: reviewId
Reviewers - primary key: reviewerId
Movies - primary key: movieId
Actors - primary key: actorId
Movie_actor - primary key: movieactorId

Connection tabellen:
Jeg har forbundet tabellerne ved at lave 4 kolonner der hver især er knyttet til reviewernes primary key. Jeg har altså *ikke* tilføjet en connection kolonne med et index i nogle af de 4 tabeller jeg bruger bare primary key'en.

INNER/LEFT JOIN movie_actor ON movieId = movieConId
INNER/LEFT JOIN movie_actor ON reviewerId= reviewerConId
INNER/LEFT JOIN movie_actor ON reviewId= reviewConId
INNER/LEFT JOIN movie_actor ON actorId= actorConId

reviewerConId	movieactorId	reviewConId	movieConId	actorConId



●connect.php

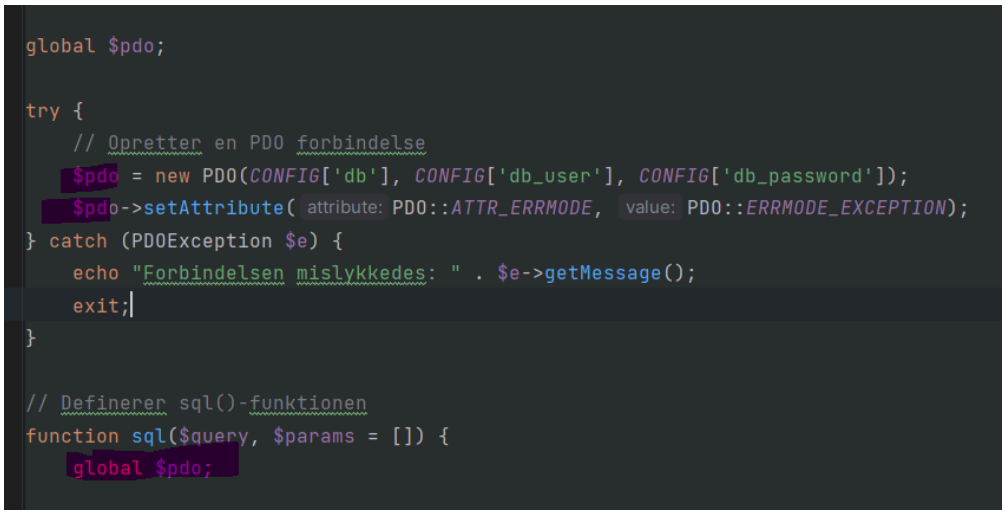
Arbejdede før på localhost, men har siden da været været inde på simply og fundet mine loginoplysinger samt oprettet en database med navnet. Derefter gik jeg ind og indtastet oplysningerne i min config.

```
const CONFIG = [
    'db' =>
'mysql:dbname=sofieroshni_dk_db_moviereviewtwo;host=mysql18.unoeuro.com;port=3306',
    'db_user' => 'sofieroshni_dk',
    'db_password' => 'R4FA35tEDcgk2xe6fGbd',
];
```

Jeg har lavet en const Config hvof jeg indtaster min databasenavn og password

-db' *indeholder oplysninger om forbindelsen til MySQL-serveren, som omfatter databasevært, port og navn.*
'db_user' *er brugernavnet for MySQL-forbindelsen.*
'db_password' *er adgangskoden.*

Global $pdo
Pdo er en inbygget klasse i php altså ikke en indbygget klasse men i indbygget variabel
Global variabel: $pdo bliver erklæret globalt, så jeg kan bruge den uden for den blok, hvor jeg har den defineret. Den bliver jo
brugt i både databaseforbindelsen og i funktionen sql().



TRY-CATCH
Try blokken bruges til at indkaplse den kode som generer fejl.
Catch blokken fanger de fejl der opstår i try-blokken- hvis fejlen opstår vil der i koden catch (PDOExepction $e) blive udført. $e er en variabel hvorpå fejlen bliver gemt.

setAttribute()
er en metode på PDO-objektet som lader mig ændrer hvordan PDO fungerer ved at sætte Attributter med metoden setAttribute()
Det første parameter i min setAttribute() er attributten jeg ønsker og ændrer, den andet parameter er den værdi jeg tildeler attributen.
I dette tilfælde har jeg skrevet ATR_ERMODE, -som mit første parameter, -det en attriubute der bestemmer hvad PDO skal gøre hvis der opstår fejl og så er det så ERMODE_EXCEPTION som er den værdi jeg tildeler attributen.
Dette betyder at pdo vil kaste en undtagelse (expection) når der opstår en fejl i stedet for at reutrnerer false.

getMessage()
getMessage() er en metode i *Exception*-klassen, der returnerer en string, som indeholder fejlinformation. Så , fanger en *PDOException* og lægger den i variablen $e. Derefter udskriver jeg en fejlmeddelelse til brugeren ved at kombinere en string ("Forbindelsen mislykkedes: ") med den specifikke fejlbesked, som hentes ved hjælp af *$e→getMessage()*.

```php
function sql($query, $params = []) {
   global $pdo;

   try {
       $stmt = $pdo->prepare($query);
       $stmt->execute($params);
       return $stmt->fetchAll(PDO::FETCH_OBJ);
   } catch (PDOException $e) {
       echo "Forespørgslen mislykkedes: " . $e->getMessage();
       return [];
   }
}
```

Function sql
Funktionen forbereder en SQL-forespørgsel ved hjælp af PDO, udfører den med de valgte parametre, og returnerer resultaterne som en array af objekter. Hvis der opstår en fejl,
håndteres fejlen med en fejlmeddelelse, og en tom array
returneres for at undgå, at programmet bryder sammen.
*ordbog:*Funktion sql defineres med to parameter

```php
function sql($query, $params = []) {
```

Funktionen sql() defineres.

```php
$query
```

Det første parameter er en streng, der indeholder den SQL-forespørgsel, der skal køres

```php
$params = []
```

Dette er en valgfri parameter, som er en array. Den indeholder de værdier, der skal indsættes i forespørgslen. Hvis der ikke sendes nogen parametre, bruges en tom array .

```php
global $pdo;
```

PDO INDE I FUNKTIONEN SQL
Variablen $pdo erklæres som global.
Dette betyder, at funktionen bruger en databaseforbindelse, som er gemt i en global variabel $pdo
. På denne måde kan vi bruge denne PDO-forbindelse inde i funktionen.

Try-block
Koden går nu ind i en try-blok. Dette bruges til at "prøve" at udføre kode, hvor der er risiko for, at noget kan gå galt (i dette tilfælde, hvis der opstår en fejl med SQL-forespørgslen). Forespørgslen (SQL-query) forberedes ved hjælp af PDO's prepare()-metode. PDO adskiller SQL-koden fra de data, der skal indsættes. Dette er vigtigt, fordi det beskytter mod SQL-injektion.

```
$stmt = $pdo->prepare($query);
```
$query: Dette er SQL-forespørgslen (strengen), som vi sender til funktionen.
$stmt: Dette er det forberedte statement (et PDO Statement-objekt), som jeg kan udføre senere.
```
$stmt->execute($params);
```
Den forberedte SQL-forespørgsel udføres ved hjælp af execute()-metoden, og parametrene (hvis nogen) sendes ind. PDO indsætter værdierne fra $params ind i de relevante pladsholdere i forespørgslen. Dette kan være en array med data, som vi har brug for i forespørgslen, f.eks. brugerinput, og det beskytter mod SQL-injektion.

Step: Resultaterne fra forespørgslen hentes. `return $stmt->fetchAll(PDO::FETCH_OBJ);`

`return $stmt->fetchAll(PDO::FETCH_OBJ);` Metoden fetchAll() henter alle resultaterne fra SQL-forespørgslen og returnerer dem som en array af objekter (hver række i resultatet bliver et objekt, hvor kolonnerne er objektets egenskaber).

`(PDO::FETCH_OBJ);` Denne konstante værdi angiver, at resultaterne skal returneres som objekter.

```
catch (PDOException $e) {
```
: Hvis der opstår en fejl under udførelsen af forespørgslen (f.eks. hvis SQL'en er ugyldig eller forbindelsen fejler)
, hopper koden til catch-blokken.
catch (PDOException $e): Hvis en fejl relateret til PDO opstår, fanger denne blok fejlen.
```
echo "Forespørgslen mislykkedes: " . $e->getMessage();
```
Fejlmeddelelsen udskrives, som forklarer præcist, hvad der gik galt.

```
return [];
```
Returnerer en tom array: I stedet for at returnere resultater, returnerer vi en tom array, da forespørgslen ikke kunne udføres

●sql forespørgsler

I nogle af sql forespørgslerne anvender jeg *LEFT JOI*N for at sikre, at alle anmeldelser fra 'reviews' tabellen vises, selvom hvis der ikke er et match i de tilsluttede tabeller.
 Dette er vigtigt, fordi jeg ønsker at se alle anmeldelser, ikke kun dem der er tilknyttet andre tabeller. Hvis jeg havde brugt INNER JOIN, ville kun de anmeldelser, jeg havde connected til i de tilsluttede tabeller, blevet vist. Det vil gøre at de anmeldelser uden forbindelse til andre af de tilsuttede tabeller vil gå tabt.

Parameterbinding, pladsholderog sql-forspørgelse
 Review.php denne side viser selve det review brugeren klikker på & afspiller skuespillerens film og filmens andre medvirkende samt

```
><a href='actor.php?actorId=$actor->actorId'>$actor->actorName</a></td>"
```

Henter id'et og sender det som et parameter via url'en til min review.php side med $_GET

```
$actor = sql("SELECT * FROM actors WHERE actorId = :actorId",
[":actorId" => $actorId]);
```

SQL for alle ser som basis sådan her ud:
På hver af siderne der linker til enten selve anmelderen, reviewet etc. har jeg i toppen lavet en

```
$actorId = $_GET["actorId"];
```

$_GET["actorId"]: Henter actorId fra URL'en.Lægger det i variabel $actorId. Anvendes til at hente den valgte skuespillers data.SQL forsprøgelserne laver jeg så ved at vælge alt fra den tabel jeg vil hente information fra og med en WHERE(betingels) at min actorId skal være lig :actorId. men sørger for at mit actorId bliver erstattet med en pladsholder, laver derfter en parameterbinding hvor pladsholderen

```
actor = sql("SELECT * FROM actors WHERE actorId = :actorId",
[":actorId" => $actorId]);
```

Henter fra min actors/movie tabel Where (betingelsen) er at actorId skal være lig med :actorværdi.

```
WHERE actorId = :actorId
```

Min WHERE begrænser resultaterne til kun at inkludere skuespilleren med den angivne actorId.

```
 [":actorId" => $actorId]);
```

Derefter laver jeg en parameterbinding ved at lægge pladsholderen ind i variablen $actorId; *(kun for at undgå SQL-indejktion)*
Når kolon : kommer foran et navn i en SQL-forespørgsel (som i dette eksempel med . :actorId), -betyder det, at jeg opretter en pladsholder (placeholder).
Derefter laver jeg en parameterbinding.
Pladsholder : actorId er en plads, som senere bliver erstattet med en rigtig værdi.
Parameterbinding: Sikrer at værdier sendes sikkert til sql forspørgelser ved at

```
$actor = $actor[0];
```

Derefter skriver jeg actorId =actor[=]; for at hente den første række i resultat sættet.

●CRUD - Gennemgår de sider hvor jeg henter og opdaterer

CREATE/INSERT -add.review.php

- laver en form med metoden post fordi jeg gerne vil sætte noget ind i databasen.

```
<form method="POST">
```

- siger hvis 'hvis sumbit-knappen er trykket bliver $_post['sumbit'] sat og koden bliver kørt

```
if (isset($_POST['submit'])) {
    $movieTitle = trim($_POST['movieTitle']) ?: null;
    $review_title = trim($_POST['review_title']) ?: null;
    $underubrik = trim($_POST['underubrik']) ?: null;
    $review_text = trim($_POST['review_text']) ?: null;
    $review_rating = (int)($_POST['review_rating'] ?? 0);
    $review_date = $_POST['review_date'] ?? null;
    $name_of_the_reviewer = trim($_POST['name_of_the_reviewer']) ?:
null;
```

- Hvis feltet er tomt eller bliver ikke er sat sættet NULL ind/0 for review_rating. ved hjælp er ??-operatoren eller ?? vil de nedstående variabler deklererers.

```
if (!$movieTitle || !$review_title || !$review_text || !$review_rating
|| !$name_of_the_reviewer || !$review_date) {
    echo "<script>alert('Alle felter skal udfyldes');</script>";
}
```

tjekker for tommefelter- Hvis de nødvendige felter er tomme vises en alertboks
Hvis !ikke $movieTitle (som jo var det der brugeren postet) || eller !$,!$,!$

```
else {
    $pdo->beginTransaction();
```

Hvis alle felter er udfyldt, starter koden en database-transaktion. Dette sikrer, at alle databaseoperationer udføres som en enkelt enhed. Hvis der opstår en fejl undervejs, kan ændringerne rulles tilbage.

'

INSERT SQL

INSERT INTO tabelnavn (movies,reviewers etc.) og skrevet mine values. til sidst har jeg connectet det i min connectiontabel ved at sige insert into movie_actor (movieCon,reviewCond,ReviewerCon)
I mit eksempel bruges ? som placeholders i prepare-metoden Det betyder, at i stedet for at skrive værdierne direkte i SQL-forespørgslen (som kunne føre til SQL-injektion), bruger jeg placeholders, der udfyldes med data, når `execute()` kaldes. *Dette er en sikkerhedsmæssig foranstaltning.*
6. indsætter anmeldelser, film, anmelderer osv.

```
$stmt1 = $pdo->prepare("INSERT INTO reviews(review_title, underubrik,
review_text, review_rating, review_date) VALUES (?, ?, ?, ?, ?)");
$stmt1->execute([$review_title, $underubrik, $review_text,
$review_rating, $review_date]);
$reviewId = $pdo->lastInsertId();

$stmt2 = $pdo->prepare("INSERT INTO movies(movieTitle) VALUES (?)");
$stmt2->execute([$movieTitle]);
$movieId = $pdo->lastInsertId();

$stmt3 = $pdo->prepare("INSERT INTO reviewers(name_of_the_reviewer)
VALUES (?)");
$stmt3->execute([$name_of_the_reviewer]);
$reviewerId = $pdo->lastInsertId();

$stmt4 = $pdo->prepare("INSERT INTO movie_actor(movieConId,
reviewConId, reviewerConId) VALUES (?, ?, ?)");
$stmt4->execute([$movieId, $reviewId, $reviewerId]);
```

UPDATE: reviews.php

```
    <form action="my_reviews.php" method="post">
        <input type="hidden" class="edit-bio-input" name="data[reviewer_id]" value="<?php echo $bio->reviewerId; ?>">
        <label>
            <input type="text" class="edit-bio-input" name="data[name_of_the_reviewer]" value="<?php echo $bio->name_of_the_reviewer; ?>">
        </label>
        <label>
            <input type="number" class="edit-bio-input" name="data[reviewer_age]" value="<?php echo $bio->reviewer_age; ?>">
        </label>
        <label>
            <input type="text" class="edit-bio-input" name="data[reviewer_email]" value="<?php echo $bio->reviewer_email; ?>">
        </label>
        <label>
            <input type="text" class="edit-bio-input" name="data[reviewer_bio]" value="<?php echo $bio->reviewer_bio; ?>">
        </label>
        <button type="submit" class="submit" name="submit">Gem</button>
    </form>
<?php else: ?>
    <p>Ingen bio fundet.</p>
<?php endif; ?>
```

Har lavet en form med action post. Samme system, laver en form med en gem knap, oppe i toppen laver jeg en “hvis submit klikkes på”

Laver en $_POST[‘data] og lægger det ind i min variabel $data.(så når brugeren har poste/indsendt data vil følgende query ske:
UPDATE TABELNAVN SET hvilke værdier(kolonner) og så WHERE reviewerId=:reviewerId
: apostrof betyder i kodning at der kan være noget derinde i stedet for. Det som der skal være inde i min :name_of_the_reviewer er $data[name_of_the_reviewer]; altså det input brugeren skriver i input feltet. Til sidst har jeg omdiregeret min side til my reviews.php

```
$data = $_POST['data'];
sql("UPDATE reviewers SET
    name_of_the_reviewer = :name_of_the_reviewer,
    reviewer_age = :reviewer_age,
    reviewer_email = :reviewer_email,
    reviewer_bio = :reviewer_bio
    WHERE reviewerId = :reviewerId", [
    ":name_of_the_reviewer" => $data["name_of_the_reviewer"],
    ":reviewer_age" => $data["reviewer_age"],
    ":reviewer_email" => $data["reviewer_email"],
    ":reviewer_bio" => $data["reviewer_bio"],
    ":reviewerId" => $data["reviewer_id"], // Sørg for at bruge kor
]);
header("Location: my_reviews.php"); // Omled til siden efter opdat

        ":reviewerId" => $data["reviewer_id"], // Sørg for
    ]);
    header("Location: my_reviews.php"); // Omled til sider
    exit;
}
```

I stedet for *insert* har jeg så skrevet *update* tabelnavn *set*, name of the reviewer.
Jeg har til sidst lavet et *bind*.

DELETE

Denne kodeblok håndterer sletning af en anmeldelse baseret på dens ID, som modtages fra URL'en.

```
if (isset($_GET['delete'])) {
    $reviewId = intval($_GET['delete']);

    $delete = sql("DELETE FROM reviews WHERE reviewId = :reviewId", [
        ":reviewId" => $reviewId
    ]);
}
```

```
<td>
    <a class="delete" href="my_reviews.php?delete=<?php echo $review->reviewId; ?>"
</td>
```

Jeg skriver i min a-tag delete efter mit spørgsmålstegn, og hvis url'en så modtager 'delete' så vil følgende ske(så vil min variabel med min sql forespørgsel hvor jeg sletter review fra ske)

```
if (isset($_GET['delete'])) {
```

Kontrollerer, om der er en "delete" parameter i URL'en. Hvis den er sat, vil koden blive udført.

```
    $reviewId = intval($_GET['delete']);
```

Sikrer, at værdien i $_GET['delete'] bliver konverteret til et heltal for at undgå SQL-injection eller fejl ved ikke-numeriske input.

Udfør sletning i databasen:

```
$delete = sql("DELETE FROM reviews WHERE reviewId = :reviewId", [
```

En SQL-forespørgsel udføres for at slette en anmeldelse i tabellen "reviews", hvor reviewId matcher det specifikke id. Den bruger en :reviewId placeholder til at beskytte mod SQL-injection.

```
    ":reviewId" => $reviewId
```

Parametre sendt til SQL. Dette sikrer, at det tilsvarende reviewId fra URL'en bruges som parameter i SQL-forespørgslen.

●Javascript, foreach+ loops

Addreview.php er en side hvor brugeren kan få lov til at skrive sin egen anmeldelse. Via $_post sendes brugerens/revieweren sin anmeldelse og både filmen+revieweren+reviewet connectes også i min connection tabel ved

6. `stars.forEach((star, index) => {`

7. laver en anonym funktion, men en addeventlistener til min stjerne(click)/stars.forEach er et loop som kører en funktion for hvert element i min liste (boostrapstjernere)
8. parametrene *star*,*index* (star er boostrapstjerner) index er "positionen af stjerne" (0, 1 eller 4)
9. i = 0, I<=INDEX; i kører op til index-af stjerner(antallet),

10. FORM VALUE: rating sendes ind i input. og input valuen sendes via min betingelse if($_POST){}

```
stars.forEach((star, index) => {
   star.addEventListener('click', () => {
        stars.forEach(star => star.classList.remove('checked'));

        for (let i = 0; i <= index; i++) {
            stars[i].classList.add('checked');
        }

        ratingInput.value = index + 1;
   });
});
```

SEARCHBAR OG API-KALD MED FETCH()

Til projektet har jeg anvendt meget lidt javascript. Jeg har lavet en search.js og koblet den til min index side hvor jeg henter jeg film fra mit API med fetch() og har lavet en søgemaskine med film, som jeg har hentet fra API.
Gennemgår search.js og
Getmovies() (en funktion som henter film)
ShowMoves[en funktion som ved foreach afspiller hver film]
Til sidst er der tilføjet en event listener til min søgeformular. Søgeordet fra brugeren gemmes i konstanten *searchTerm* i inputfeltet. *En if-else-betingelse* kontrollerer, om brugeren har indtastet noget, og aktiverer *getMovies()* -funktionen, hvis det er tilfældet.

getMovies() denne funktion henter filmene.
Fetch(url,options) Jeg fetcher min url med de definerede options (som er objektet der indeholder autorisationen og headers) *.then(res⇒res.json)()* når data bliver modtaget håndteres de som en response ved at kalde .json(), får vi svaret som json-data *.then(data⇒{showMovies(data.results)}}* Her bliver data.results (listen med film) sendt til funktionen *showMovies()* . Funktionen *showMovies()* kan så bruge denne liste til at vise hver film på din side.

 showMovies(data) denne funktion afspiller filmene.
mvoieContainer.innerHtml ='' tømmes
derefter laver jeg en forEach hvor movies er mit parameter. Min movies er et objekt med udvalgte informationer jeg hentede fra via fetch hentede fra mit data-array .Derefter laver jeg et div element og afspiller min informationer. Jeg sørger for at jeg skipper de film uden billeder Jeg tilføjer til sidst en mine film(div) til min html struktur (movieContainer).

searchForm.addEventListeners
Jeg har til min Form har jeg tilføjet en Addeventlistener.
Den "værdi" brugeren søgeord har jeg lagt ind i min konstant SearchTerm skriver inde i mit input felt. Derefter laver jeg en if else med betingelsen, at hvis brugeren sørger så skal getMovies funktion ske.

●fremadrettet plan for moviereview

Hvad der fremover skal ske for din side og gør de uklare ting i din synopsis mere klar.
Det ville være smart hvis jeg bagom scenen lavede en insert. Specielt når jeg har en mange til mange tabel:
lave et tomt array der hedder actors og så lave en når man trykker på film
kunne man jeg sige $get[movieId];
skrive teams som et tomt array, og så sige hver gang at
 $_POST['actors] as actor{
sql("insert into moviecon (movieConid,ActorConId) VALUES (:movieConId,:actorConId)",
["*movieConId⇒$_post[movieId], actorConId⇒[':actorConId']"])
}
Jeg ved ikke om det var smartere hvis jeg tog udgangspunkt i min connectiontabel.

TODO:

- UPLOAD BILLEDER I DIN MOVIES
- LAV EN KORT FORKLARING DER HANDLER OM HVORDAN DU KUNNE HAVDE HAFT TAGET UDGANGSPUNKT I DIN CONNECTION-TABEL OG INDSAT MOVIES PÅ DEN MÅDE.
- GÅ IGENNEM HVORDAN MAN SIKRER SIG AT DER IKKE OPRETTES EN NY FILM HVERGANG MAN OPRETTER EN NY ANMELDELSE.
- UPLOAD PÅ GITHUB+MOVIEREVIEW
- SKRIV I HEAD.PHP $REVIEWER→REVIEWER_IMAGE;
- LAV TALEKORT HVOR DU SNAKKER OM FEJL I DIN DATABASE
- FIND UD AF HVORFOR DIN SEARCH.JS IKKE DUR+SEARCHBAR, LAV TALEKORT OG KOM MED EN MULIGVIS LØSNING.
- LAV EN LOGIN-SIDE
- SKRIV TALEKORT TIL HVAD DER SKAL SKE FOR SIDEN MOVIEREVIEW