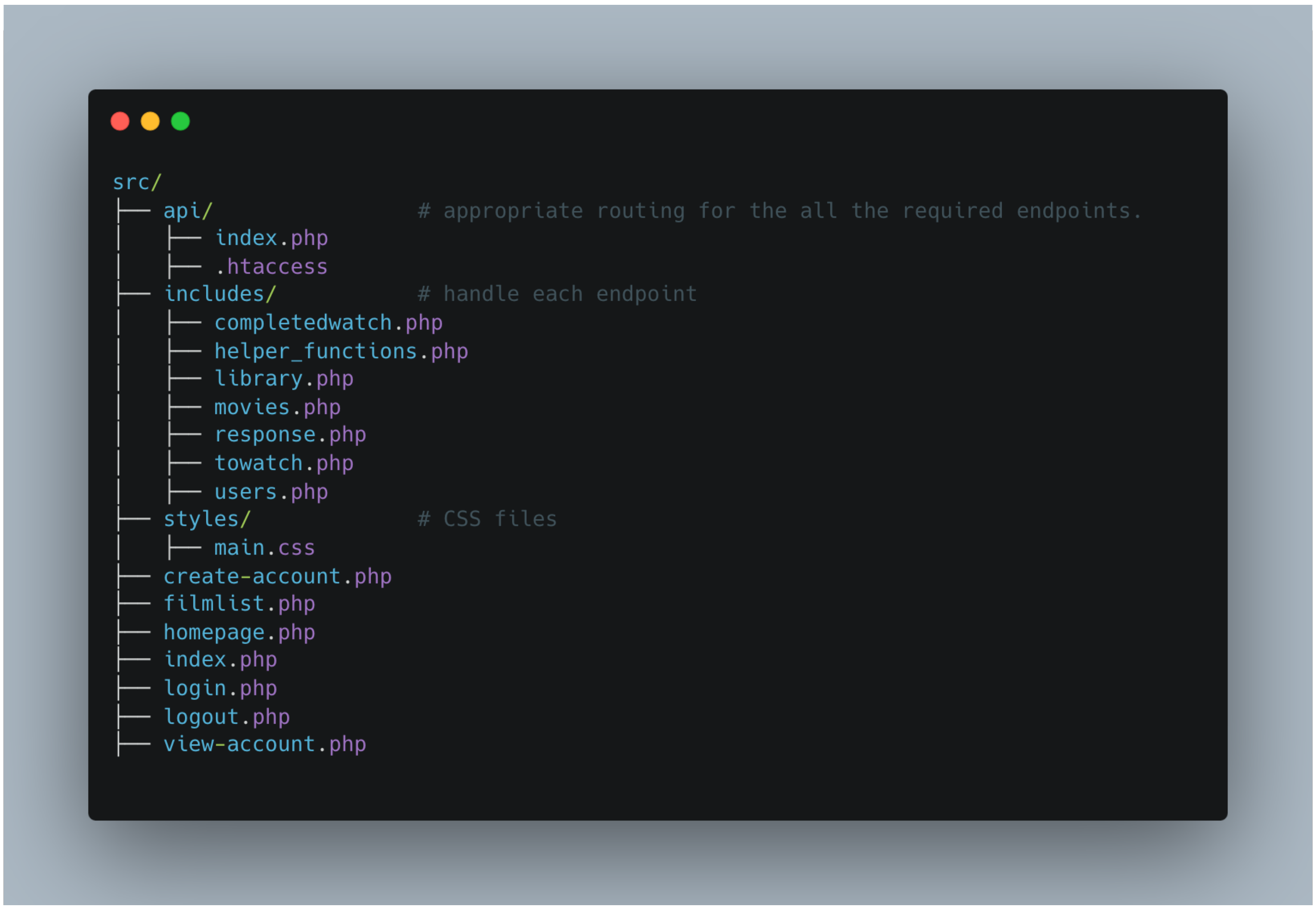
Assignment 2 Code Snippets

- Assignment 2's Structure:



```
src/
├── api/                # appropriate routing for the all the required endpoints.
│   ├── index.php
│   ├── .htaccess
├── includes/           # handle each endpoint
│   ├── completedwatch.php
│   ├── helper_functions.php
│   ├── library.php
│   ├── movies.php
│   ├── response.php
│   ├── towatch.php
│   ├── users.php
├── styles/             # CSS files
│   ├── main.css
├── create-account.php
├── filmlist.php
├── homepage.php
├── index.php
├── login.php
├── logout.php
├── view-account.php
```

- api/ index.php:

```
<?php

if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {

    header("Access-Control-Allow-Origin: *");
    header("Access-Control-Allow-Methods: *");
    // header("Access-Control-Allow-Headers: Content-Type, Authorization, 
X-Requested-With, X-API-KEY");
    header("Access-Control-Allow-Headers: *");

    http_response_code(200);
    exit();
}

require_once "../includes/library.php";
require_once "../includes/response.php";
require_once "../includes/movies.php";
require_once "../includes/towatch.php";
require_once "../includes/completedwatch.php";
require_once "../includes/users.php";

//Get the request method from the server array
$method = $_SERVER['REQUEST_METHOD'];
$uri = $_SERVER['REQUEST_URI'];
$uri = parse_url($uri);

// Define the consistent BASE URL for your API
define('__BASE__', '/~litran/3430/assn/assn2-tlinhh10102003/api/');
$endpoint = explode('/', trim(str_replace(__BASE__, '', $uri['path']),
'/'));

// Check if the endpoint is set correctly
if (count($endpoint) < 1 || empty($endpoint[0])) {
    json_response(404, ['error' => 'Endpoint not found']);
    exit;
}

switch ($endpoint[0]) {
    case 'movies':
        handleMoviesRouting($method, $endpoint);
        break;
    case 'towatchlist':
        handleToWatchListRouting($method, $endpoint);
        break;
    case 'completedwatchlist':
        handleCompletedWatchListRouting($method, $endpoint);
        break;
    case 'users':
        handleUsersRouting($method, $endpoint);
        break;
    default:
        json_response(404, ['error' => 'Endpoint not found']);
}
?>
```

- api/ .htaccess:

```
<Limit GET POST PUT DELETE PATCH OPTIONS>
  Require all granted
</Limit>
#enable apache rewrite engine
RewriteEngine On

# Turn on Auth Header
RewriteCond %{HTTP:Authorization} ^(.*)
RewriteRule .* - [e=HTTP_AUTHORIZATION:%1]

#set the base directory for rewrite to the folder your api is in
RewriteBase /~litran/3430/assn/assn2-tlinhh10102003/api

#deliver the folder of file directly if it exists
```

```
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f

#push every request to index.php
RewriteRule ^(.+)$ index.php [QSA,L]
```

- includes/completedwatch.php:

```php
<?php
require_once "library.php";
require_once "response.php";
require_once "helper_functions.php";

function handleCompletedWatchListRouting($method, $endpoint) {
    if ($method === "GET") {
        if(isset($endpoint[1]) && $endpoint[1] === 'entries') {
            if(isset($endpoint[2]) && is_numeric($endpoint[2])) {
                if(isset($endpoint[3]) && $endpoint[3] === "times-
watched") {
                    getCompletedWatchTimesWatched($endpoint[2]);
                }
                elseif(isset($endpoint[3]) && $endpoint[3] === "rating") {
                    getCompletedWatchRating($endpoint[2]);
                }
            }
            else {
                getAllCompletedWatchEntries();
            }
        }
    }
    elseif($method === "POST") {
        if (isset($endpoint[1]) && $endpoint[1] === 'entries') {
            addCompletedWatchListEntry();
        } else {
            json_response(404, ['error' => 'Invalid endpoint']);
        }
    }
    elseif($method === "PATCH") {
        // /completedwatchlist/entries/{id}
        if(isset($endpoint[1]) && $endpoint[1] === 'entries' &&
isset($endpoint[2]) && is_numeric($endpoint[2])) {
            // /completedwatchlist/entries/{id}/times-watched
            if(isset($endpoint[3]) && $endpoint[3] === "times-watched") {
                updateCompletedWatchTimesWatched($endpoint[2]);
            }
            // /completedwatchlist/entries/{id}/rating
            elseif(isset($endpoint[3]) && $endpoint[3] === "rating") {
                updateCompletedWatchRating($endpoint[2]);
            }
        }
    }
    elseif($method === "DELETE" && isset($endpoint[1]) && $endpoint[1] ===
'entries' && isset($endpoint[2]) && is_numeric($endpoint[2])) {
        deleteEntryFromCompletedWatch($endpoint[2]);
    }
    else {
        json_response(405, ['error' => 'This method is not allowed. We are
unable to respond to this request.']);
    }
}

//GET /completedwatchlist/entries
function getAllCompletedWatchEntries(){
    $user_id = validateApiKey();

    $pdo = connectdb();
    // filter minimum rating filter if max_rating is in the url
    if (isset($_GET['max_rating']) && $_GET['max_rating'] === 'true') {
        $query = "select movie_id, title, MAX(COMPLETED_WATCH_LIST.rating)
as max_rating from COMPLETED_WATCH_LIST
                join MOVIES on MOVIES.id = COMPLETED_WATCH_LIST.movie_id
where user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
    }
    else {
        $query = "select cwl.id, cwl.user_id, cwl.rating as userRating,
m.id as movieID, m.title, m.overview, m.rating, m.poster, cwl.notes,
cwl.times_watched, cwl.date_last_watched
            from COMPLETED_WATCH_LIST cwl
            join MOVIES m ON m.id = cwl.movie_id
            where user_id = ?
            ORDER BY cwl.date_last_watched DESC";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
    }
    $cwl = $stmt->fetchAll(PDO::FETCH_ASSOC);
    json_response(200, $cwl);
}

//GET /completedwatchlist/entries/{id}/times-watched
function getCompletedWatchTimesWatched($id){
    $user_id = validateApiKey();

    $pdo = connectdb();
    $query = "select times_watched from COMPLETED_WATCH_LIST where id = ?
and user_id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$id, $user_id]);
    $times_watched = $stmt->fetch(PDO::FETCH_ASSOC);
    if(!$times_watched) {
        json_response(404, ['error' => 'Entry not found in the
database']);
    }
    json_response(200, $times_watched);
}
```

```
//GET /completedwatchlist/entries/{id}/rating
function getCompletedWatchRating($id){
    $user_id = validateApiKey();

    $pdo = connectdb();
    $query = "select rating from COMPLETED_WATCH_LIST where id = ? and
user_id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$id, $user_id]);
    $rating = $stmt->fetch(PDO::FETCH_ASSOC);
    if(!$rating) {
        json_response(404, ['error' => 'Entry not found in the
database']);
    }
    json_response(200, $rating);
}

function addCompletedWatchListEntry() {
    $user_id = validateApiKey();

    // get data from JSON body
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data) {
        $data = $_POST;
    }

    // verify required data to insert in table
    if (!isset($data['movie_id']) || !isset($data['rating']) ||
!isset($data['notes']) || !isset($data['date_watched']) ||
!isset($data['times_watched'])) {
        json_response(400, ['error' => 'Completed: Missing required
fields']);
        return;
    }

    $pdo = connectdb();

    try {
        // use transaction to complete multiple taks: insert data and
update rating in movies
        $pdo->beginTransaction();
        $query = "insert into COMPLETED_WATCH_LIST (user_id, movie_id,
rating, notes, date_watched, date_last_watched, times_watched)
                  VALUES (?, ?, ?, ?, ?, NOW(), ?)";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id, $data['movie_id'], $data['rating'],
$data['notes'], $data['date_watched'], $data['times_watched']]);

        // update rating
        recomputeAvgRating($pdo, $data['movie_id'], $data['rating']);

        // commit if everything went well
        $pdo->commit();
        json_response(201, ['message' => 'New entry added to completed
watchlist successfully']);
    } catch (Exception $e) {
        // rollbakc  if an error occurred
        $pdo->rollBack();
        json_response(500, ['error' => 'Failed to add entry', 'details' =>
$e->getMessage()]);
    }
}


//PATCH /completedwatchlist/entries/{id}/rating
function updateCompletedWatchRating($id){
    $user_id = validateApiKey();
    if ($user_id) {
        $movie_id = getMovieID($user_id, $id);
    }

    // get data from JSON body
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['new_rating']) || !is_numeric($data['new_rating'])) {
        json_response(400, ['error' => 'Invalid or missing new_rating
field']);
        return;
    }

    $pdo = connectdb();

    try {
        // use transaction to complete multiple taks
        $pdo->beginTransaction();
        //check if entry exists
        $query = "select rating from COMPLETED_WATCH_LIST where id = ? and
user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$id, $user_id]);
        $old_rating = $stmt->fetch(PDO::FETCH_ASSOC);

        // if no entry exists
        if (!$old_rating) {
            json_response(404, ['error' => 'Entry not found']);
            return;
        }
        $updateQuery = "update COMPLETED_WATCH_LIST set rating = ? where
id = ? and user_id = ?";
        $stmt = $pdo ->prepare($updateQuery);
        $stmt->execute([$data['new_rating'], $id, $user_id]);

        // update rating in movies table
        recomputeAvgRatingAfterUpdate($pdo, $movie_id,
$old_rating['rating'], $data['new_rating']);

        // commit if everything went well
        $pdo->commit();
        json_response(200, ['message' => 'Rating updated successfully',
'rating' => $data['new_rating']]);

    } catch (Exception $e) {
        // rollback if an error occurred
        $pdo->rollBack();
        json_response(500, ['error' => 'Failed to updating rating of given
entry', 'details' => $e->getMessage()]);
    }
}

// PATCH /completedwatchlist/entries/{id}/times-watched
function updateCompletedWatchTimesWatched($id){
    $user_id = validateApiKey();
    if ($user_id) {
        $movie_id = getMovieID($user_id, $id);
    }

    $pdo = connectdb();

    try {
        // use transaction to complete multiple task
        $pdo->beginTransaction();
        //check if entry exists
        $query = "select times_watched from COMPLETED_WATCH_LIST where id
= ? and user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$id, $user_id]);
        $times_watched = $stmt->fetch(PDO::FETCH_ASSOC);

        if(!$times_watched) {
            json_response(400, ['error' => 'Entry not found in the
database']);
        }

        $updated_times_watched = $times_watched['times_watched'] + 1;

        // update complted watch list table
        $updateQuery = "update COMPLETED_WATCH_LIST set times_watched = ?,
date_last_watched = NOW()
                        where id = ? and user_id = ?";
        $stmt = $pdo->prepare($updateQuery);
        $stmt->execute([$updated_times_watched, $id, $user_id]);

        $pdo->commit();
        json_response(200, ['message' => '"times_watched" and
"date_last_watched" updated successfully', 'times_watched' =>
$updated_times_watched]);
    }
    catch(Exception $e) {
        $pdo->rollBack();
        json_response(500, ['error' => 'Failed to update times watched',
'details' => $e->getMessage()]);
    }
}

// DELETE /completedwatchlist /entries/{id}
function deleteEntryFromCompletedWatch($id) {
    $user_id = validateApiKey();

    // get data from json body
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data) {
        $data = $_POST;
    }
    if(!isset($data['movie_id'])) {
        json_response(400, ['error' => 'Missing required field']);
        return;
    }
    $pdo = connectdb();
    // check if entry exists
    $pdo = connectdb();
    $checkQuery = "select * from COMPLETED_WATCH_LIST where id = ? and
user_id = ?";
    $stmt = $pdo->prepare($checkQuery);
    $stmt->execute([$id, $user_id]);
    $exists = $stmt->fetchColumn();

    if($exists) {
        // delete
        $query = "delete from COMPLETED_WATCH_LIST where movie_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$data['movie_id']]);
        json_response(200, ['message' => 'Movie deleted successfully']);
    }
    elseif(!$exists) {
        json_response(400, ['error' => 'Entry not found in the
database']);
    }
}

//recompute rating
function recomputeAvgRating($pdo, $movie_id, $new_rating) {
    $query = "select rating, rating_count from MOVIES where id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$movie_id]);
    $movie = $stmt->fetch(PDO::FETCH_ASSOC);

    // if the movie exists
    if ($movie) {
        $old_rating = $movie['rating'];
        $old_rating_count = $movie['rating_count'];

        if ($old_rating_count === 0) {
            $new_avg_rating = $new_rating;
            $new_rating_count = 1;
        } else {
            $new_rating_count = $old_rating_count + 1;
            $new_avg_rating = (($old_rating * $old_rating_count) +
$new_rating) / $new_rating_count;
        }

        // update the movies table
        $updateQuery = "update MOVIES set rating = ?, rating_count = ?
where id = ?";
        $updateStmt = $pdo->prepare($updateQuery);
        $updateStmt->execute([$new_avg_rating, $new_rating_count,
$movie_id]);
    }
}

function recomputeAvgRatingAfterUpdate($pdo, $movie_id, $old_rating,
$new_rating) {
    $query = "select rating, rating_count from MOVIES WHERE id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$movie_id]);
    $movie = $stmt->fetch(PDO::FETCH_ASSOC);
    if($movie) {
        $old_avg_rating = $movie['rating'];
        $count = $movie['rating_count'];

        if ($count > 1) {
            $new_avg_rating = (($old_avg_rating * $count) - $old_rating +
$new_rating) / $count;
        } else {
            $new_avg_rating = $new_rating;  // if no rating exists before
new rating = new avg rating
        }

        $updateQuery = "update MOVIES set rating = ?, rating_count = ?
where id = ?";
        $updateStmt = $pdo->prepare($updateQuery);
        $updateStmt->execute([$new_avg_rating, $count, $movie_id]);
    }
}
?>
```

- includes/helper_functions.php:

```
<?php
function getUserId($apiKey) {
    $pdo = connectdb();
    $query = "select id from USERS where api_key = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$apiKey]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);
    // return user ID if found, null if not found
    return $user ? $user['id'] : null;
}

function getApiKey() {
    // chekc if the API key is provided in the request headers
    if (isset($_SERVER['HTTP_X_API_KEY'])) {
        return $_SERVER['HTTP_X_API_KEY'];
    }
    return null;
}

function validateApiKey() {
    $api_key = getApiKey();
    if (!$api_key) {
        json_response(403, ['error' => 'An API key is required!']);
        exit;
    }

    $user_id = getUserId($api_key);
    if (!$user_id) {
        json_response(403, ['error' => 'Invalid API key!']);
        exit;
    }

    return $user_id;
}

function getMovieID($user_id, $id) {
    $pdo = connectdb();
    $query = "select movie_id from COMPLETED_WATCH_LIST where id = ? and
user_id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute(([$id, $user_id]));
    $movie_id = $stmt->fetch(PDO::FETCH_ASSOC);;
    // return movie ID if found, null if not found
    return $movie_id ? $movie_id['movie_id'] : null;
}
?>
```

- includes/library.php:

```
<?php
// Get the acutal document and webroot path for virtual directories
$direx = explode('/', getcwd());
define('DOCROOT', "/$direx[1]/$direx[2]/"); // /home/username/
define('WEBROOT', "/$direx[1]/$direx[2]/$direx[3]/");
//home/username/public_html

function connectdb()
{
  // Load configuration as an array.
  $config = parse_ini_file(DOCROOT . "pwd/config.ini");
  $dsn =
"mysql:host=$config[domain];dbname=$config[dbname];charset=utf8mb4";

  try {
    $pdo = new PDO($dsn, $config['username'], $config['password'], [
      PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
      PDO::ATTR_EMULATE_PREPARES   => false,
    ]);
  } catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
  }

  return $pdo;
}
```

- includes/movies.php:

```
<?php
require_once "library.php";
require_once "response.php";
require_once "helper_functions.php";

function handleMoviesRouting($method, $endpoint) {
    if ($method === "GET") {
        // GET & /movies/{id}
        if (isset($endpoint[1]) && is_numeric($endpoint[1])) {
            // GET & /movies/{id}/rating
            if(isset($endpoint[2]) && $endpoint[2] === "rating") {
                // retrieve rating detail of a specific movie
                getMovieRating($endpoint[1]);
            }
            else {
                // retrieve detail for specific movie with given id
                getMovie($endpoint[1]);
            }
        }
        // GET & /movies/search
        elseif(isset($endpoint[1]) && $endpoint[1] ==="search") {
            searchMovies();
        }
        // GET & /movies/ - retrieve all movies
        else {
            getAllMovies();
        }
    } else {
        // if method is not GET
        json_response(405, ['error' => 'This method is not allowed. We are
unable to respond to this request.']);
    }
```

```
}

function getAllMovies() {
    $pdo = connectdb();

    $title = isset($_GET['title']) ? $_GET['title'] : null;
    if($title) {
        $query = "select id, title, overview, rating, poster, release_date
from MOVIES where title like ? LIMIT 1000";
        $stmt = $pdo->prepare($query);
        $stmt->execute(['%' . $title . '%']);
    }
    else {
        $query = "select id, title, overview, rating, poster, release_date
from MOVIES LIMIT 1000";
        $stmt = $pdo->query($query);
    }
    $movies = $stmt->fetchAll(PDO::FETCH_ASSOC);
    json_response(200, $movies);
}

function getMovie($id) {
    $pdo = connectdb();
    $query = "select * from MOVIES where id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$id]);
    $movie = $stmt->fetch(PDO::FETCH_ASSOC);

    // if movie with given id exists
    if ($movie) {
        json_response(200, $movie);
    } else {
        json_response(404, ['error' => 'Movie not found']);
    }
}

function getMovieRating($id) {
    $pdo = connectdb();
    $query = "select rating from MOVIES where id = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$id]);
    $rating = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($rating) {
        json_response(200, ['rating' => $rating['rating']]);
    } else {
        json_response(404, ['error' => 'Movie not found']);
    }
}

function searchMovies() {
    // $api_key = getApiKey();
    // if (!$api_key) {
    //     json_response(403, ['error' => 'An API key is required']);
    //     return;
    // }

    // $user_id = getUserId($api_key);
    // if (!$user_id) {
    //     json_response(403, ['error' => 'Invalid API key']);
    //     return;
    // }

    // Get search query from the request
    $query = isset($_GET['q']) ? $_GET['q'] : '';
    error_log("Received query: " . $query);
    $pdo = connectdb();
    $stmt = $pdo->prepare("SELECT * FROM MOVIES WHERE title LIKE ? LIMIT 
30");
    $stmt->execute(["%$query%"]);
    $movies = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if ($movies) {
        json_response(200, ['results' => $movies]);
    } else {
        json_response(404, ['message' => 'No movies found']);
    }
}
?>
```

- includes/response.php:

```


<?php
function json_response($statusCode, $response_data = []) {
    // set HTTP status code
    http_response_code($statusCode);

    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Origin: *");
    header("Access-Control-Allow-Methods: *");
    header("Access-Control-Allow-Headers: *");

    // Encode response data
    $json_encode_data = json_encode($response_data, JSON_PRETTY_PRINT);
    header('Content-Length: ' . strlen($json_encode_data));
    echo $json_encode_data;
    exit();
}
?>
```

- includes/towatch.php:

```
<?php
require_once "library.php";
require_once "response.php";
require_once "helper_functions.php";

function handleToWatchListRouting($method, $endpoint) {
    if($method === "GET"  && $endpoint[1] === "entries") {
        getAllEntries();
    }
    elseif($method === "POST"  && $endpoint[1] === "entries") {
        if(isset($endpoint[2]) && is_numeric($endpoint[2]) &&
isset($endpoint[3]) && $endpoint[3] === "watched")
        {
            markAsWatched($endpoint[2]);
        }
        else {
            addToWatchListEntry();
        }
    }
    elseif($method === "PUT"  && $endpoint[1] === "entries" &&
isset($endpoint[2]) && is_numeric($endpoint[2])) {
        updateToWatchListEntry($endpoint[2]);
    }
    elseif($method === "PATCH" && isset($endpoint[2]) &&
isset($endpoint[3]) && $endpoint[3] === "priority") {
        updateToWatchListPriority($endpoint[2]);
    }
    elseif($method === "DELETE" && isset($endpoint[2])) {
        deleteToWatchListEntry($endpoint[2]);
    }
    else {
        // other method
        json_response(405, ['error' => 'This method is not allowed. We are
unable to respond to this request.']);
    }
}

// get method
function getAllEntries(){
    $user_id = validateApiKey();

    $pdo = connectdb();

    $priority = isset($_GET['priority']) ? (int)$_GET['priority'] : null;
// filter towatch by priority if it appears in the url
    if($priority) {
        $query = "SELECT twl.id, twl.user_id, twl.priority, m.id as
movieID, m.title, m.overview, m.rating, twl.notes
                  FROM TO_WATCH_LIST twl
                  JOIN MOVIES m ON m.id = twl.movie_id
                  WHERE user_id = ? AND twl.priority <= ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id, $priority]);
    } else { // else just return all movies in the to watch list
        $query = "SELECT twl.id, twl.user_id, twl.priority, m.id as
movieID, m.title, m.overview, m.rating, twl.notes, m.poster
                  FROM TO_WATCH_LIST twl
                  JOIN MOVIES m ON m.id = twl.movie_id
                  WHERE user_id = ?
                  ORDER BY twl.priority";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
    }

    $twl = $stmt->fetchAll(PDO::FETCH_ASSOC); // fetchAll to get all
matching entries
    json_response(200, $twl);
}

// post methiod
function addToWatchListEntry() {
    $user_id = validateApiKey();
    // get data from JSON body
    $data = json_decode(file_get_contents('php://input'),
JSON_PRETTY_PRINT);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['movie_id']) || !isset($data['priority']) ||
!isset($data['notes'])) {
        json_response(400, ['error' => 'Add to Watch List: Missing
required fields']);
        return;
    }

    $pdo = connectdb();

    // Check if the movie is already in the user's watchlist
    $checkWatchList = "SELECT COUNT(*) FROM TO_WATCH_LIST WHERE user_id =
? AND movie_id = ?";
    $stmt = $pdo->prepare($checkWatchList);
    $stmt->execute([$user_id, $data['movie_id']]);
    $watchlist_exist = $stmt->fetchColumn();

    $checkCompletedList = "SELECT COUNT(*) FROM COMPLETED_WATCH_LIST WHERE
user_id = ? AND movie_id = ?";
    $stmt = $pdo->prepare($checkCompletedList);
    $stmt->execute([$user_id, $data['movie_id']]);
    $completedwatch_list = $stmt->fetchColumn();

    if ($watchlist_exist || $completedwatch_list) {
        json_response(409, ['error' => 'Movie is already in your list']);
        return;
    }
    $query = "insert into TO_WATCH_LIST (user_id, movie_id, priority,
notes) values (?, ?, ?, ?)";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$user_id, $data['movie_id'], $data['priority'],
$data['notes']]);

    json_response(201, ['message' => 'Entry added to watchlist']);
}

function markAsWatched($id) {
    // Retrieve API key and validate it
    $user_id = validateApiKey();

    // Get the request data from the JSON body (like notes and rating)
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data) {
        $data = $_POST;
    }

    // Default the note to an empty string if not provided
    $note = isset($data['note']) ? trim($data['note']) : "";
    $rating = isset($data['rating']) ? $data['rating'] : null;

    $pdo = connectdb();
    try {
        $pdo->beginTransaction(); // Start a transaction to ensure
atomicity

        // Check if the movie exists in the "To Watch" list for the
current user
        $checkQuery = "SELECT * FROM TO_WATCH_LIST WHERE id = ? AND
user_id = ?";
        $stmt = $pdo->prepare($checkQuery);
        $stmt->execute([$id, $user_id]);
        $entry = $stmt->fetch(PDO::FETCH_ASSOC);

        if (!$entry) {
            // If the movie isn't in the "To Watch" list, send an error
            json_response(404, ['error' => 'Movie not found in To Watch
List']);
            return;
        }

        // Remove the movie from the "To Watch" list
        $deleteQuery = "DELETE FROM TO_WATCH_LIST WHERE id = ? AND user_id
= ?";
        $stmt = $pdo->prepare($deleteQuery);
        $stmt->execute([$id, $user_id]);

        // Insert the movie into the "Completed List"
        $insertQuery = "INSERT INTO COMPLETED_WATCH_LIST (user_id,
movie_id, notes, rating, date_last_watched) VALUES (?, ?, ?, ?, NOW())";
        $stmt = $pdo->prepare($insertQuery);
        $stmt->execute([$user_id, $entry['movie_id'], $note, $rating]);
        // Commit the transaction if everything is successful
        $pdo->commit();

        // Send a successful response
        json_response(200, ['message' => 'Movie marked as watched
successfully']);
    } catch (Exception $e) {
        // Rollback the transaction if an error occurred
        $pdo->rollBack();
        json_response(500, ['error' => 'Failed to mark entry as watched',
'details' => $e->getMessage()]);
    }
}


function updateToWatchListEntry($id) {
    $user_id = validateApiKey();

    $pdo = connectdb();

    // Check if the entry exists
    $queryCheck = "SELECT * FROM TO_WATCH_LIST WHERE id = ? AND user_id =
?";
    $stmt = $pdo->prepare($queryCheck);
    $stmt->execute([$id, $user_id]);
    $exists = $stmt->fetchColumn();

    // Get data from the JSON body
    $data = json_decode(file_get_contents('php://input'),
JSON_PRETTY_PRINT);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['movie_id']) || !isset($data['priority'])) {
        json_response(400, ['error' => 'Update Watch List: Missing
required fields']);
        return;
    }

    $notes = isset($data['notes']) ? $data['notes'] : "";

    // If entry exists, update it
    if($exists) {
        $query = "UPDATE TO_WATCH_LIST
                  SET movie_id = ?, priority = ?, notes = ?
                  WHERE id = ? AND user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$data['movie_id'], $data['priority'], $notes, $id,
$user_id]);
        json_response(200, ['message' => 'Entry updated successfully']);
    } else {
        // If entry doesn't exist, create a new entry
        $query = "INSERT INTO TO_WATCH_LIST (user_id, movie_id, priority,
notes)
                  VALUES (?, ?, ?, ?)";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id, $data['movie_id'], $data['priority'],
$notes]);
        json_response(201, ['message' => 'Entry added successfully']);
    }
}

// patch method
function updateToWatchListPriority($id) {
    $user_id = validateApiKey();

    // get data from JSON body
    $data = json_decode(file_get_contents('php://input'),
JSON_PRETTY_PRINT);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['priority'])) {
        json_response(400, ['error' => 'Update Watch List Priority:
Missing required fields']);
        return;
    }

    // check if entry exists
    $pdo = connectdb();
    $checkQuery = "select * from TO_WATCH_LIST where id = ? and user_id =
?";
    $stmt = $pdo->prepare($checkQuery);
    $stmt->execute([$id, $user_id]);
    $exists = $stmt->fetchColumn();

    // if exist then update priority
    if($exists) {
        $query = "update TO_WATCH_LIST set priority = ? where user_id = ?
and id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$data['priority'], $user_id, $id]);
        json_response(200, ['message' => 'Entry priority updated
successfully']);
    }
    // no entry exists then error response
    else {
        json_response(404, ['error' => 'Entry not found in the
database']);
    }
}

// delete method
function deleteToWatchListEntry($id) {
    $user_id = validateApiKey();
    // get data from JSON body
    $data = json_decode(file_get_contents('php://input'),
JSON_PRETTY_PRINT);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['movie_id'])) {
        json_response(400, ['error' => 'Delete Watch List: Missing
required fields']);
        return;
    }

    // check if entry exists
    $pdo = connectdb();
    $checkQuery = "select * from TO_WATCH_LIST where id = ? and user_id =
?";
    $stmt = $pdo->prepare($checkQuery);
    $stmt->execute([$id, $user_id]);
    $exists = $stmt->fetchColumn();

    // if exist then delete appropriate movie
    if($exists) {
        $query = "delete from TO_WATCH_LIST where movie_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$data['movie_id']]);
        json_response(200, ['message' => 'Movie deleted successfully']);
    }
    // // no entry exists then error response
    // else {
    //     json_response(404, ['error' => 'Entry not found in the
database']);
    // }
}

?>
```

- includes/users.php:

```


<?php
require_once "library.php";
require_once "response.php";
require_once "helper_functions.php";

function handleUsersRouting($method, $endpoint) {
    if($method === "GET"){
        if (isset($endpoint[1]) && is_numeric($endpoint[1]) &&
isset($endpoint[2]) && $endpoint[2] === "stats"){
            getUserStat($endpoint[1]);
        }
    }
    elseif($method === "POST")
    {
        if(isset($endpoint[1]) && $endpoint[1] === "session")
        {
            userSession();
        }
    }
    else {
        json_response(405, ['error' => 'This method is not allowed. We are
unable to respond to this request.']);
    }
}

function getUserStat($id) {
    $user_id = validateApiKey();

    $pdo = connectdb();
    $stats = []; // initialize empty array store watching stats

    // filter for most-watched movies if requested
    if (isset($_GET['most_watched']) && $_GET['most_watched'] === 'true')
{
        $query = "select m.title, cwl.times_watched
                  from COMPLETED_WATCH_LIST cwl
                  join MOVIES m ON cwl.movie_id = m.id
                  where cwl.user_id = ?
                  order by cwl.times_watched DESC
                  limit 1";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
        $stats['most_watched_movies'] = $stmt->fetch(PDO::FETCH_ASSOC);
    }
    else {
        // get total movies watched
        $query = "select COUNT(*) AS total_movies from
COMPLETED_WATCH_LIST where user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
        $stats['total_movies_watched'] = $stmt->fetchColumn();

        // get total watched times
        $query = "select SUM(times_watched) AS total_watched_times from
COMPLETED_WATCH_LIST where user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
        $stats['total_watched_times'] = $stmt->fetchColumn();

        // get avarage rating
        $query = "select AVG(rating) AS avg_Rating from
COMPLETED_WATCH_LIST where user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
        $stats['average_rating'] = $stmt->fetchColumn();
        // get number of movies plan to watch
        $query = "select COUNT(*) AS plan_to_watch from TO_WATCH_LIST
where user_id = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$user_id]);
        $stats['plan_to_watch'] = $stmt->fetchColumn();
    }
    json_response(200, $stats);
}

function userSession(){
    // get data from json body
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data) {
        $data = $_POST;
    }

    if (!isset($data['username']) || !isset($data['password'])) {
        json_response(400, ['error' => 'Username and password are
required1']);
        return;
    }

    $pdo = connectdb();
    // fetch user from the database
    $query = "select id, password, api_key from USERS where username = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$data['username']]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    //user is found in the db and password is correct
    //compare entered password vs password of the user stored in the db
    if(!$user || !password_verify($data['password'], $user['password'])) {
        json_response(401, ["error" => "Invalid username or password"]);
        return;
    }
    else {
        $api_key = $user['api_key'];
        $user_id = $user['id'];

        json_response(200, [
            'Your API key' => $api_key,
            'user_id' => $user_id // Include user_id in the response
        ]);
    }
}
?>
```

- styles/main.css:

```css
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/
@import url('https://fonts.googleapis.com/css2?
family=Unlock&display=swap');
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}

table {
    border-collapse: collapse;
    border: solid black;
    letter-spacing: 0.1em;
    width: 90%;
    margin: 1em auto;
}
```

```
th, td {
    border: 0.1em solid black;
    padding: 1em;
    text-align: center;
}
td {
    display: table-cell;
    vertical-align: inherit;
    padding: 0.5em 1em 0.5em 0.5em;
}
th {
    display: table-cell;
    vertical-align: inherit;
    font-weight: bold;
    text-align: center;
    padding: 1em;
    background-color: var(--2nd-color-primary);
    color: var(--color-primary);
}
tr {
    display: table-row;
    vertical-align: inherit;
}

li {
    display: list-item;
    padding: 0.8em;
}

footer {
    position: fixed;
    bottom: 0;
    max-width: 1600px;
    width: 100vw;
    margin-left: auto;
    margin-right: auto;
    min-width: 600px;
    margin: auto;
    text-align: center;
    font-family: "Unlock";
}

strong {
    font-weight: 700;
}
em {
    font-style: italic;
}

:root {
    --bgcolor: #dccdb8;
    --color-primary: #b64931;
    --2nd-color-primary: #ffc567;
    --3rd-color-primary: #00995e;
}

.error {
    color: red;
}
.highlight {
    color: deeppink;
    font-weight: 600;
}
/*##########################################*/

/*#########################################

HomePage

#############################################*/
#homepage {
    background-image: url('../1.png');
    background-size: cover;
    background-attachment: fixed;
    background-repeat: no-repeat;
    height: auto;
}
#nav-bar {
    font-family: "Unlock";
    display: grid;
    width: 90%;
    margin: auto;
    grid-template-columns: 1.5fr 2fr 1fr;
    margin-top: 0.5em;
    padding: 1em 0.5em;
    border: solid var(--color-primary);
    border-radius: 2em;
    background-color: var(--color-primary);
    justify-items: auto;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    position: relative;
    z-index: 2;
}
#nav-bar img,
#filmlist img {
    position: absolute;
    width: 1.8em;
    margin: -24px 0 0 0.5em;
    transition: transform 0.3s ease;
}
#nav-bar img:hover,
#filmlist img:hover,
#login img:hover,
#create-acc img:hover,
#view-acc img:hover,
#api img:hover {
    transform: rotate(10deg) scale(1.1);
}
```

```
#header {
    font-weight: 600;
    font-size: 3em;
    margin-left: 0.5em;
    text-shadow: 2px 2px black;
}
#header a {
    text-decoration: none;
    color: var(--2nd-color-primary);
    width: 20%;
    transition: all 0.3s ease;
}
#header a:hover {
    color: var(--3rd-color-primary);
}
#nav-bar form {
    display: flex;
    justify-content: center;
    align-items: center;
}
#nav-bar form input {
    font-family: "Unlock";
    border-radius: 0.5em;
    padding: 0.5em;
}
#nav-bar form input:focus {
    outline: none;
    border-color: var(--color-primary);
    box-shadow: 0 0 8px rgba(108, 123, 159, 0.5);
}
#nav-bar form button {
    padding: 0;
    margin: 0;
    border: none;
    color: var(--color-primary);
    background-color: var(--color-primary);
}
#nav-bar form i {
    position: absolute;
    margin: 0 0 0 12em;
}

#options {
    text-shadow: 2px 2px black ;
    display: grid;
    grid-template-columns: repeat(4, auto);
    align-content: center;
}
#options a {
    margin: 0.5em 1em;
    color: var(--2nd-color-primary);
    text-decoration: none;
    border: solid var(--color-primary);
    border-radius: 1em;
    text-align: center;
    padding: 0.5em;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    transition: background-color 0.3s, transform 0.2s;
}
#options a:hover {
    background-color: #6c7b9f;
    border-color: #6c7b9f;
    transform: scale(1.2);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
}
.movies-list,
#list,
#endpoints {
    width: 80%;
    background-color: rgba(249, 242, 234, 0.9) ;
    border: solid rgba(249, 242, 234, 0.9);
    border-radius: 1em;
    margin: 1.5em auto;
    box-shadow: 0 0 15px rgba(0,0,0,0.1);
}
.movies-list h1,
#filmlist h1,
#api h1 {
    font-family: "Unlock";
    width: 50%;
    font-weight: 600;
    font-size: x-large;
    text-align: center;
    border-radius: 0.6em;
    color: var(--color-primary);
    background-color: var(--2nd-color-primary);
    margin: 0.5em auto 0.5em auto;
    padding: 0.5em;
}

#homepage p {
    width: 20%;
    background-color: rgba(249, 242, 234, 0.9);
    border: solid black;
    border-radius: 1em;
    padding: 1em;
    margin: 1em auto;
    text-align: center;
    font-weight: 600;
}

/*###########################################

Film List page

##############################################*/
#filmlist,
#login,
```

```css
#create-acc,
#api {
    background-image: url("../bg.png");
    background-size: cover;
    background-attachment: fixed;
    background-repeat: no-repeat;
    height: auto;
}
#filmlist #header {
    font-family: "Unlock";
    display: grid;
    width: 90%;
    margin: auto;
    margin-top: 0.2em;
    padding: 0.3em;
    padding-left: 0.5em;
    border: solid var(--color-primary);
    border-radius: 0.6em;
    background-color: var(--color-primary);
}
#filmlist #header a {
    text-decoration: none;
    color: var(--2nd-color-primary);
}
#filmlist #header a:hover {
    color: var(--3rd-color-primary);
}
#filmlist img,
#login img,
#create-acc img,
#view-acc img,
#api img {
    position: absolute;
    width: 1.8em;
    margin: -24px 0 0 6.1em;
}

#list h1 {
    font-family: "Unlock";
}

.page-section {
    background-color: rgba(249, 242, 234, 0.9);
    text-align: center;
    margin: 1em 0;
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 0.5em;
    padding: 1em;
    flex-wrap: wrap;
}
.page-section a {
    padding: 0.2em;
    color: var(--2nd-color-primary);
    text-decoration: none;
    border-radius: 0.2em;
    transition: background-color 0.3s;
}
.page-section a:first-of-type,
.page-section a:last-of-type {
    color: black;
    font-weight: 600;
}

.page-section a:hover {
    background-color: #6c7b9f;
    border-color: #6c7b9f;
    color: white;
}

.page-section .active {
    font-weight: 600;
    background-color: var(--color-primary);
    color: white;
}

/*###########################################

Create account page

############################################*/
#success-message p {
    color: green;
    text-align: center;
    background-color: rgba(249, 242, 234, 0.9);
    border: solid rgba(249, 242, 234, 0.9);
    border-radius: 0.5em;
    width: 40%;
    margin: 1em auto 5em auto;
    padding: 0.2em;
    font-weight: 600;
}
#sign-up {
    font-family: "Unlock";
    background-color: rgba(249, 242, 234, 0.9);
    border: solid black;
    border-radius: 0.5em;
    width: 30%;
    margin: 10em auto 0 auto;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
}
#sign-up span {
    display: flex;
    justify-content: center;
    margin-top: 0;
    text-align: center;
}
```

```
/*########################################

Log In page

###########################################*/
#login-form,
#acc-info {
    font-family: "Unlock";
    background-color: rgba(249, 242, 234, 0.9);
    border: solid black;
    border-radius: 0.5em;
    width: 30%;
    margin: 12em auto;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
}
#login-form h1,
#sign-up h1,
#acc-info h1 {
    font-family: "Unlock";
    font-size: large;
    text-align: center;
    color: white;
    background-color: var(--2nd-color-primary);
    border-bottom: solid black;
    border-radius: 0.4em;
    padding: 0.5em;
    margin-bottom: 1em;
    text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);
}
#login-form form input,
#sign-up form input {
    display: flex;
    justify-items: center;
    margin: 0.5em auto 0 auto;
    padding: 0.5em;
    border-color: black;
    border-radius: 1em;
    width: 50%;
    transition: border-color 0.3s;
}
#login-form form input:focus,
#sign-up form input:focus {
    border-color: var(--color-primary);
}
#login-form form button,
#sign-up form button {
    font-family: "Unlock";
    display: flex;
    justify-content: center;
    align-items: center;
    color: var(--3rd-color-primary);
    background-color: var(--2nd-color-primary);
    border: solid var(--2nd-color-primary);
    border-radius: 0.5em;
    padding: 0.5em;
    margin: 0.5em auto 1.2em auto;
    transition: background-color 0.3s, transform 0.2s;
    cursor: pointer;
}
#login-form form button:hover,
#sign-up form button:hover {
    color: var(--2nd-color-primary);
    background-color: var(--color-primary);
    border-color: var(--color-primary);
    transform: scale(1.2);
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}
#login #header,
#create-acc #header,
#view-acc #header,
#api #header {
    display: grid;
    width: 90%;
    margin: auto;
    font-family: "Unlock";
    margin-top: 0.2em;
    padding: 0.3em;
    padding-left: 0.5em;
    border: solid var(--color-primary);
    border-radius: 0.6em;
    background-color: var(--color-primary);
    justify-items: auto;
}
label {
    margin: 0.5em;
    display: block;
    text-align: center;
}

/*##########################################

View Account page

#############################################*/
#view-acc {
    background-image: url("../bg.png");
    background-size: auto;
    height: auto;
}
#acc-info form {
    display: flex;
}
#acc-info form button,
#acc-info form a {
    font-family: "Unlock";
    color: var(--3rd-color-primary);
    background-color: var(--2nd-color-primary);
```

```css
    border: solid var(--2nd-color-primary);
    border-radius: 0.5em;
    padding: 0.5em;
    margin: 1em auto;
    transition: background-color 0.3s, transform 0.2s;
    cursor: pointer;
}
#acc-info form a {
    text-align: center;
    text-decoration: none;
    font-size: 0.82em;
    width: 20%;
    padding: 0.5em;
    transition: background-color 0.3s, transform 0.2s;
    cursor: pointer;
}
#acc-info form button:hover,
#acc-info form a:hover {
    color: var(--2nd-color-primary);
    background-color: var(--color-primary);
    border-color: var(--color-primary);
    transform: scale(1.2);
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}
.success-message {
    color: green;
    margin: 10px 0;
    text-align: center;
}
/*###########################################

API Documentation page

############################################*/
#api h1 {
    margin-top: 0.5em;
    margin-bottom: 0;
}
#api h2 {
    border-radius: 0.6em;
    text-align: center;
    font-weight: 600;
    width: 40%;
    color: var(--color-primary);
    background-color: rgba(249, 242, 234, 0.9);
    padding: 0.5em;
    margin: auto;
}
#api p {
    margin: 1em;
}
```

- create-account.php:

```php
<?php
include "./includes/library.php";
$pdo = connectdb();
$username = $_POST['username'] ?? " ";
$email = $_POST['email'] ?? " ";
$password = $_POST['password'] ?? " ";
$errors = [];
$successMessage = "";
if ($_SERVER['REQUEST_METHOD'] === 'POST')
{
    // if username is empty
    if(empty($username)) {
        $errors['username'] = "You must provide a username";
    }
    else {
        //validate username uniqueness
        $query = "select id from USERS where username = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$username]);
        if ($stmt->rowCount() > 0) {
            $errors['username'] = "Username already existed";
        }
    }

    //validate email
    if (empty(filter_var($email, FILTER_VALIDATE_EMAIL))) {
        $errors['email'] = "Please provide a valid email address";
    }

    //validate pwd
    if(empty($password)) {
        $errors['password'] = 'Please enter your password';
    }
    elseif (strlen($password) < 13  || !preg_match('/[0-9]/', $password)
|| !preg_match('/[~`!@#$%^&*()\-_=+{}\[\]|\\:;"<>,.?\/]/', $password) ||
!preg_match('/[A-Z]/', $password)) {
        $errors['password'] = "Password must be at least 12 characters
long, minimum of 1 numeric character,
                    minimum of 1 special character, and minimum of 1
uppercase letter";
    }

    if (empty($errors)) {
        $password_hash = password_hash($password, PASSWORD_BCRYPT);

        $api_key = bin2hex(random_bytes(16));
        // Insert the user into the database
        $query = "insert into USERS (username, email, password, api_key,
api_date) VALUES (?, ?, ?, ?, NOW())";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$username, $email, $password_hash, $api_key]);
        // Set the success message after successful account creation
        $successMessage = "Account created successfully! Your API key is:
$api_key";
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create Account</title>
    <link rel="stylesheet" href="./styles/main.css">
</head>

<body id="create-acc">
    <div id="header">
        <a href="homepage.php"> MovieLand </a>
        <img src="./tv.png" alt="TV retro style"/>
    </div>

    <div id="sign-up">
        <h1> Create Account </h1>
        <form method="post" >
            <label for="username">Username: </label>
            <input type="text" id="username" name="username" value="<?=
$username ?>"><br>
            <span class="error <?= !isset($errors['name']) ? 'hidden' : ''
?>">
                <?= $errors['username'] ?? '' ?>
            </span>

            <label for="email">Email: </label>
            <input type="text" id="email" name="email" value="<?= $email ?
>"><br>
            <span class="error <?= !isset($errors['email']) ? 'hidden' :
'' ?>">
                <?= $errors['email'] ?? '' ?>
            </span>

            <label for="password">Password: </label>
            <input type="password" id="password" name="password"><br>
            <span class="error <?= !isset($errors['password']) ? 'hidden'
: '' ?>">
                <?= $errors['password'] ?? '' ?>
            </span>

            <button id="submit" type="submit" name="submit"><strong>Create
Account</strong></button>
        </form>
    </div>

    <div id="success-message">
        <?php
        if (!empty($successMessage)) {
            echo "<p>$successMessage</p>";
        }
        ?>
    </div>

    <footer>&copy; MovieLand, Inc. 2024</footer>
</body>
</html>
```

- filmlist.php:

```
<?php
include "./includes/library.php";

$pdo = connectdb();

// divide results into pages
$results_per_page = 30;

// current page or default = 1
$page = isset($_GET['page']) && is_numeric($_GET['page']) ?
(int)$_GET['page'] : 1;

// starting point for each page
$start_from = ($page - 1) * $results_per_page;

$total_movies_query = "select count(*) AS total from MOVIES";
$total_movies_stmt = $pdo->query($total_movies_query);
$total_movies = $total_movies_stmt->fetch(PDO::FETCH_ASSOC)['total'];
$total_pages = ceil($total_movies / $results_per_page);

// fetch the movies for the current page
$query = "SELECT id, title, overview, rating
          FROM MOVIES
          LIMIT :start_from, :results_per_page";

$stmt = $pdo->prepare($query);
$stmt->bindValue(':start_from', (int)$start_from, PDO::PARAM_INT);
$stmt->bindValue(':results_per_page', (int)$results_per_page,
PDO::PARAM_INT);

if ($stmt->execute()) {
    $movies = $stmt->fetchAll(PDO::FETCH_ASSOC);
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>FILM LIST</title>
    <link rel="stylesheet" href="./styles/main.css">
    <script src="https://kit.fontawesome.com/a8c4a04983.js"
```

```
crossorigin="anonymous"></script>
</head>

<body id="filmlist">
    <div id="header">
        <a href="homepage.php"> MovieLand </a>
        <img src="./tv.png" alt="TV retro style"/>
    </div>

    <div id="list">
        <h1>MOVIES LIST</h1>
        <?php if (!empty($movies)): ?>
            <table id="movies">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Title</th>
                        <th>Overview</th>
                        <th>Rating</th>
                    </tr>
                </thead>
                <tbody>
                    <?php foreach($movies as $movie): ?>
                        <tr>
                            <td><?= ($movie['id']) ?></td>
                            <td><?= ($movie['title']) ?></td>
                            <td><?= ($movie['overview']) ?></td>
                            <td><?= ($movie['rating']) ?></td>
                        </tr>
                    <?php endforeach; ?>
                </tbody>
            </table>
        <?php else: ?>
            <p>No movies found for this page.</p>
        <?php endif; ?>
    </div>

    <div class="page-section">
        <?php if ($page > 1): ?>
            <a href="filmlist.php?page=<?= $page - 1 ?>">Previous</a>
        <?php endif; ?>

        <?php for ($i = 1; $i <= $total_pages; $i++): ?>
            <a href="filmlist.php?page=<?= $i ?>" class="<?= $i == $page ?
'active' : '' ?>"><?= $i ?></a>
        <?php endfor; ?>

        <?php if ($page < $total_pages): ?>
            <a href="filmlist.php?page=<?= $page + 1 ?>">Next</a>
        <?php endif; ?>
    </div>

    <footer>&copy; MovieLand, Inc. 2024</footer>
```

```
</body>
</html>
```

- homepage.php:

```
<?php
require_once "./includes/library.php";
$movies = [];
$search = isset($_POST['search']) ? $_POST['search'] : " ";
if($_SERVER['REQUEST_METHOD'] === "POST")
{
    $search = strtolower($search);
    if($search) {
        $pdo = connectdb();
        $query = "SELECT id, title, overview, rating FROM MOVIES WHERE
LOWER(title) LIKE ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute(['%' . strtolower($search) . '%']);
        $movies = $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
    else {
        // Return all movies if no search term is provided
        $query = "SELECT id, title, overview, rating, poster FROM MOVIES";
        $stmt = $pdo->query($query);
    }

}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HomePage</title>
    <link rel="stylesheet" href="./styles/main.css">
    <script src="https://kit.fontawesome.com/a8c4a04983.js"
crossorigin="anonymous"></script>
</head>

<body id = "homepage">
    <div id="nav-bar">
        <div id="header">
            <a href="homepage.php"> MovieLand </a>
            <img src="./tv.png" alt="TV retro style"/>
        </div>
        <div id="options">
            <a href="login.php">LOG IN</a>
            <a href="create-account.php">SIGN UP</a>
            <a href="filmlist.php">FILM LIST</a>
            <a href="index.php">API DOCUMENTATION</a>
        </div>

        <form method="POST">
```

```
            <button id="submit" type="submit" name="submit">
<strong>Search</strong></button>
            <i id="search-glass" class="fa-solid fa-magnifying-glass"
style="color: #b64931"></i>
            <input type="text" name="search" id="search" value="<?=
$search ?>" />
        </form>
    </div>

    <?php if (!empty($movies)): ?>
    <div class="movies-list">
        <h1>SEARCH RESULTS</h1>

        <table>
        <thead>
        <th>Title</th>
        <th>Overview</th>
        <th>Rating</th>
        </thead>
        <tbody>
            <?php foreach ($movies as $movie):
            if (!empty($search)) {
                if($search && stripos($movie['title'], $search) !== FALSE)
                {
                    //if search term is found in the string, return new
string which replace original search term with the highlighted one
                    if($search && stripos($movie['title'], $search) !==
FALSE) {
                        $highlightedOverview = str_ireplace($search, "
<span class='highlight'>$search</span>", $movie['overview']);
                        $highlightedTitle = str_ireplace($search, "<span
class='highlight'>$search</span>", $movie['title']);
                    }
                    else {
                        $highlightedTitle = str_ireplace($search, "<span
class='highlight'>$search</span>", $movie['title']);
                    }
                }
            }
                ?>
            <tr>
            <td><?= $highlightedTitle ?></td>
            <td><?= $highlightedOverview ?></td>
            <td><?= ($movie['rating']) ?></td>
            </tr>
            <?php endforeach; ?>
        </tbody>
        </table>
    </div>

    <?php elseif ($_SERVER['REQUEST_METHOD'] === "POST"): ?>
    <p>No results found for "<?= ($search); ?>"</p>
    <?php endif; ?>
    <footer>&copy; MovieLand, Inc. 2024</footer>
</body>
</html>
```

- index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>API Documentation</title>
    <link rel="stylesheet" href="./styles/main.css">
</head>
<body id="api">
    <div id="header">
        <a href="homepage.php"> MovieLand </a>
        <img src="./tv.png" alt="TV retro style"/>
    </div>

    <h1>API ENDPOINTS</h1>

    <h2>Welcome to the MovieLand API, where allows you to search for
movies, manage your to-watch movie list or completed watchlist,
        as well as access to your account data. You'll find more detail as
below</h2>
    <div id="endpoints">

        <p><strong><em>1. /movies Endpoint</em></strong></p>
        <ul>
            <li><strong>GET /movies/</strong> - Retrieve a list of all
movies.</li>
            <li><strong>GET /movies/{id}</strong> - Retrieve detailed
information about a specific movie.</li>
            <li><strong>GET /movies/{id}/rating</strong> - Retrieve rating
of a specific movie given its id.</li>
        </ul>

        <hr>

        <p><strong><em>2. /towatchlist Endpoint</em></strong></p>
        <ul>
            <li><strong>GET /towatchlist/entries</strong> - Retrieve all
entries on the to-watch list.</li>
            <li><strong>POST /towatchlist/entries</strong> - Add a movie
to the to-watch list.</li>
            <li><strong>PUT /towatchlist/entries/{id}</strong> - Replace a
movie in the to-watch list.</li>
            <li><strong>PATCH /towatchlist/entries/{id}/priority</strong>
-  Updates the user's priority for the appropriate movie.</li>
            <li><strong>DELETE /towatchlist/entries/{id}</strong> - Delete
the appropriate movie from the user's towatch list..</li>
```

```
        </ul>

        <hr>

        <p><strong><em>3. /completedwatchlist Endpoint</em></strong></p>
        <ul>
            <li><strong>GET /completedwatchlist/entries</strong> - Retrieve all entries on the completed watchlist.</li>
            <li><strong>GET /completedwatchlist/entries/{id}/times-watched</strong> - Requires an api key and returns the number of times the user has watched the given movie.</li>
            <li><strong>GET /completedwatchlist/entries/{id}/rating</strong> - requires an api key and returns the user's rating for this specific movie.</li>
            <li><strong>POST /completedwatchlist/entries </strong> - requires an api key and all other data necessary for the completedWatchList table, validates then inserts the data. It should also recompute and update the rating for the appropriate movie.</li>
            <li><strong>PATCH /completedwatchlist/entries/{id}/rating</strong> - requires an api key and new rating and updates the rating for the appropriate movie in the completedWatchList table, then recalculates the movie's rating and updates the movies table. </li>
            <li><strong> PATCH /completedwatchlist/entries/{id}/times-watched </strong> - requires an api key and increments the number of times watched and updates the last date watched of the appropriate movie. </li>
            <li><strong> DELETE /completedwatchlist /entries/{id}</strong> - requires and api key and movieID and deletes the appropriate movie from the completedWatchList. </li>
        </ul>

        <hr>

        <p><strong><em>4. /users Endpoint</em></strong></p>
        <ul>
            <li><strong>GET & /users/{id}/stats</strong> - returns basic watching stats for the provided user. You can chose the stats, but you should have at least 4. e.g. total time watched, average score, planned time to watch, etc.</li>
            <li><strong>POST & /users/session</strong> - accepts a username and password, verifies these credentials and returns the corresponding API key. (You can mostly steal this logic from your login page above, just generate json responses instead)</li>
        </ul>

    </div>
    <footer>&copy; MovieLand, Inc. 2024</footer>
</body>
</html>
```

- login.php:

```
<?php
session_start();
include "./includes/library.php";
include "./includes/response.php";
$pdo = connectdb();
$username = $_POST['username'] ?? " ";
$password = $_POST['password'] ?? " ";
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if(empty($username)) {
        $errors['username'] = "You must provide a username";
    }

    // fetch user from the database
    $query = "select id, password, api_key from USERS where username = ?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$username]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    //user is found in the db and password is correct
    //compare entered password vs password of the user stored in the db
    if($user && password_verify($password, $user['password'])) {
        //store user id and api key in the session array
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['api_key'] = $user['api_key'];

        //redirect to view-account
        header("Location: view-account.php");
        exit();
    }
    else {
        json_response(401, ["error" => "Invalid username or password"]);
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title> Login </title>
    <link rel="stylesheet" href="./styles/main.css">
</head>

<body id="login">
    <div id="header">
        <a href="homepage.php"> MovieLand </a>
        <img src="./tv.png" alt="TV retro style"/>
    </div>

    <div id="login-form">
        <h1> Login </h1>
        <form method="post" >
            <label for="username">Username: </label>
            <input type="text" id="username" name="username" value="<?=
$username ?>"/>
            <span class="error <?= !isset($errors['name']) ? 'hidden' : ''
?>">
                <?= $errors['username'] ?? '' ?>
            </span>
            <br>

            <label for="password">Password: </label>
            <input type="password" id="password" name="password"><br>

            <button id="submit" type="submit" name="submit">
<strong>Submit</strong></button>
        </form>
    </div>
    <footer>&copy; MovieLand, Inc. 2024</footer>
</body>
</html>
```

- logout.php:

```
<?php
session_start();
session_destroy();
header("Location: homepage.php");
exit();
?>
```

- view-account.php:

```
<?php
session_start();
include "./includes/library.php";
$pdo = connectdb();

//if user is not logged in redirect them back to log in page
if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit;
}

//fecth user detail
$query = "select username, email, api_key from USERS where id = ?";
$stmt = $pdo->prepare($query);
$stmt->execute([$_SESSION['user_id']]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);

//regenerate api key if requested
if($_SERVER['REQUEST_METHOD'] === 'POST') { // regenerate button is
submitted
```

```
    $new_api_key = bin2hex(random_bytes(16));
    //update user api key in the db
    $query = "update USERS set api_key = ?, api_date = NOW() where id =
?";
    $stmt = $pdo->prepare($query);
    $stmt->execute([$new_api_key, $_SESSION['user_id']]);

    //update api key in the session
    $_SESSION['api_key'] = $new_api_key;

    // Redirect with a success message
    header("Location: view-account.php?
success=API+key+successfully+regenerated+!");
    exit();
}
$timeout_duration = 600; // 10'

if (isset($_SESSION['LAST_ACTIVITY']) && (time() -
$_SESSION['LAST_ACTIVITY']) > $timeout_duration) {
    session_unset();
    session_destroy();
    header("Location: homepage.php?timeout=true");
    exit();
}

// Update the last activity time
$_SESSION['LAST_ACTIVITY'] = time();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title> View Account </title>
    <link rel="stylesheet" href="./styles/main.css">
</head>

<body id="view-acc">
    <div id="header">
        <a href="homepage.php"> MovieLand </a>
        <img src="./tv.png" alt="TV retro style"/>
    </div>

    <div id="acc-info">
        <h1> Account Details </h1>
        <ol>
            <li>Username: <?php echo ($user['username']); ?></li>
            <li>Email: <?php echo ($user['email']); ?></li>
            <li>API Key: <?php echo ($_SESSION['api_key']); ?></li>
        </ol>
        <form method="post" action="">
            <button type="submit">Regenerate API Key</button>
            <a href="logout.php">Log out</a>
        </form>
        <?php if (isset($_GET['success'])): ?>
        <p class="success-message"><?php echo
htmlspecialchars($_GET['success']); ?></p>
        <?php endif; ?>
    </div>
    <footer>&copy; MovieLand, Inc. 2024</footer>
</body>
</html>
```