



WEB PROGRAMMING 2 - TP

LARAVEL - INSTALLATION

- I. Install any web server having Apache and Mysql
 - Wamp, Xampp, EasyPhp, etc.
- II. Install composer :
 - Laravel uses composer to manage dependencies
 - <https://getcomposer.org/download/>
 - After installation, type in the command prompt '*Composer*'



```
D:\Program Files\XAMPP>composer

Composer version 1.10.10 2020-08-03 11:35:19
```

LARAVEL - INSTALLATION

III. Install Laravel :

- Go to your web server root folder
- Run the command to create your first Laravel project

```
Composer create-project --prefer-dist laravel/laravel mylaravel
```

- Start Laravel services by executing

```
php artisan serve
```

```
D:\Program%20Files\Wamp64\www\mylaravel>php artisan serve  
Laravel development server started: http://127.0.0.1:8000  
[Fri Sep 18 08:59:31 2020] PHP 7.4.0 Development Server (http://127.0.0.1:8000) started
```

- Open the URL in the browser

EXERCISE 1- SIGNUP

The purpose is to create a fully functional Signup form.

In order, we will divide our work into multiple steps.

Welcome to our site

Last name

First name

Gender ☐ Male ☐ Female

Email

Password

Confirm Password

EXERCISE 1- SIGNUP

Step 1:

- Create the HTML form

Step 2:

- Create corresponding route: localhost:8000/Signup

Step 3:

- Create user controller
- When user clicks the Register button, we need to display a message
'Welcome Firstname, Lastname'

EXERCISE 1- SIGNUP

Step 4:

- Set database connection string
- Create user Model and DB table
- Add needed methods (for CRUD operations)

Step 5:

- Set the Signup logic:
 - Verify email is not already taken
 - Encrypt password before adding the record to the DB
 - Display a message to the user

EXERCISE 2- GET ALL USERS

Step1:

We need to display the list of users we added in exercise 1 following the below grid.

Last Name	First Name	Email	Gender
Smith	John	john.smith@ua.edu.lb	Male
Smith	Jane	jane.smith@ua.edu.lb	Female

EXERCISE 2- GET ALL USERS

Step 2:

Add to the above grid , a simple search.

User can search by

- Name
- Email

Name

Email

Search

Last Name	First Name	Email	Gender	
Smith	John	john.smith@ua.edu.lb	Male	Edit Delete
Smith	Jane	jane.smith@ua.edu.lb	Female	Edit Delete

EXERCISE 2- GET ALL USERS

Step 3:

When user clicks on a record, he will be redirected to the user details page.

Name	John Smith
Email	john.smith@ua.edu.lb
Gender	Male

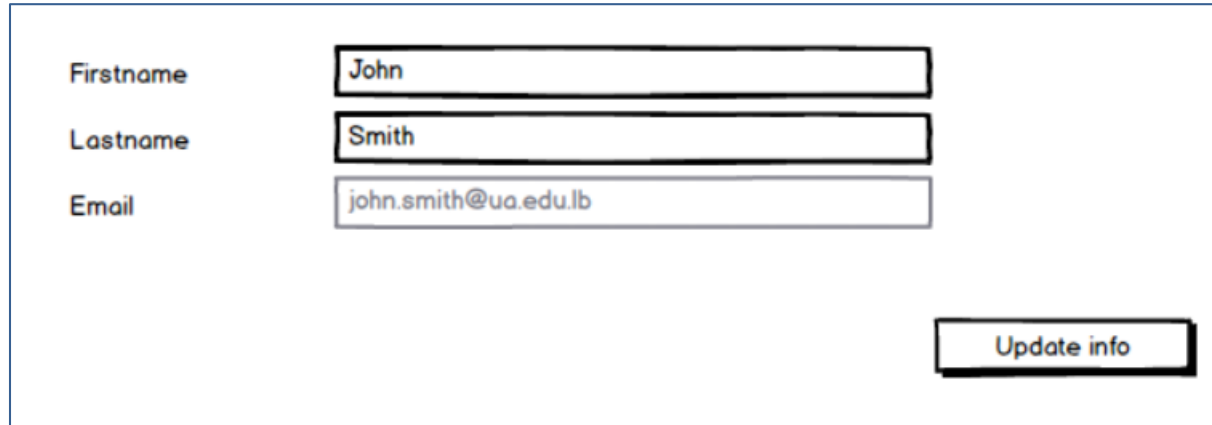
Back

EXERCISE 2- GET ALL USERS

Step 4:

When user clicks on *edit*, he will be redirected to the below screen to update his info.

When user clicks on *delete*, show a confirmation popup before deleting the intended user.



A form for updating user information. It contains three input fields for 'Firstname', 'Lastname', and 'Email'. The 'Firstname' field contains 'John', the 'Lastname' field contains 'Smith', and the 'Email' field contains 'john.smith@ua.edu.lb'. There is an 'Update info' button at the bottom right.

Firstname	<input type="text" value="John"/>
Lastname	<input type="text" value="Smith"/>
Email	<input type="text" value="john.smith@ua.edu.lb"/>

EXERCISE 3- MOVIE APPLICATION

We need to build a movie review application using Laravel where users can:

- browse through list of movies
- search for movies
- check movie details page
- leave reviews and rating



EXERCISE 3- MOVIE APPLICATION

In this web app, we support two roles:

Admin can:

- Add/ update/ delete movies
- Delete reviews

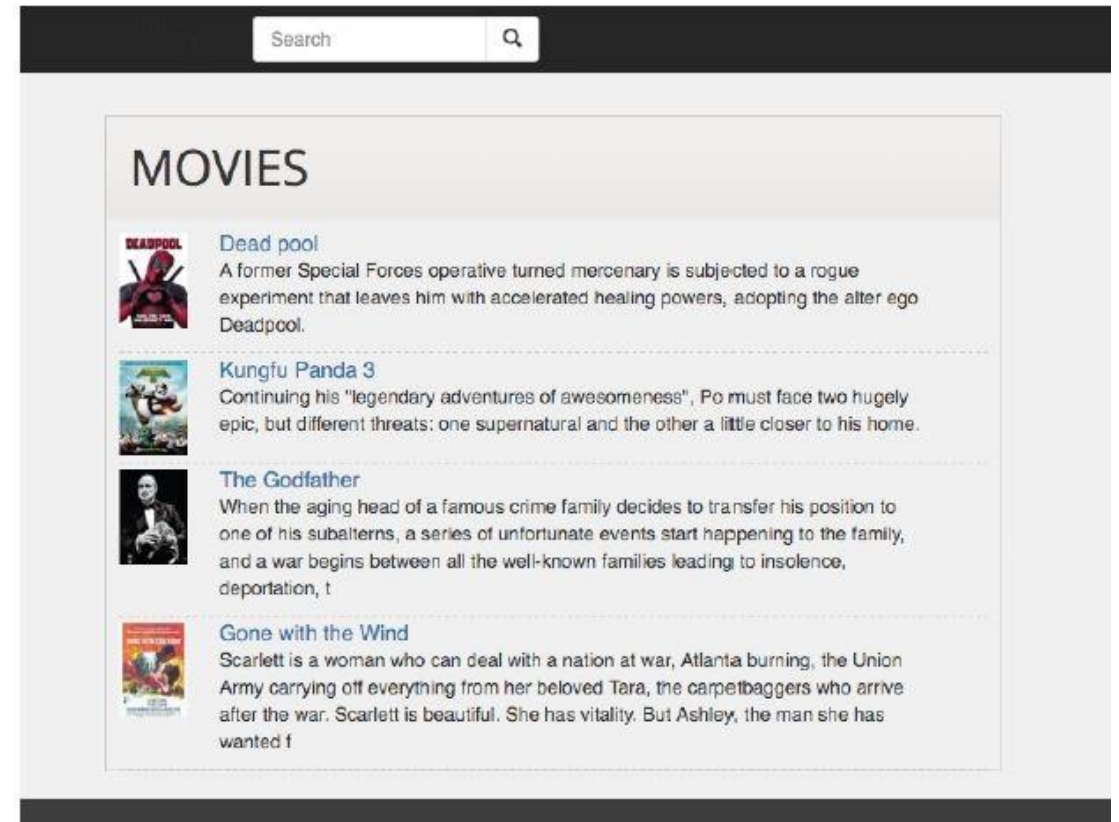
User can :

- Browse and search for movies
- Check movie details
- Add review

EXERCISE 3- MOVIE APPLICATION

Landing page

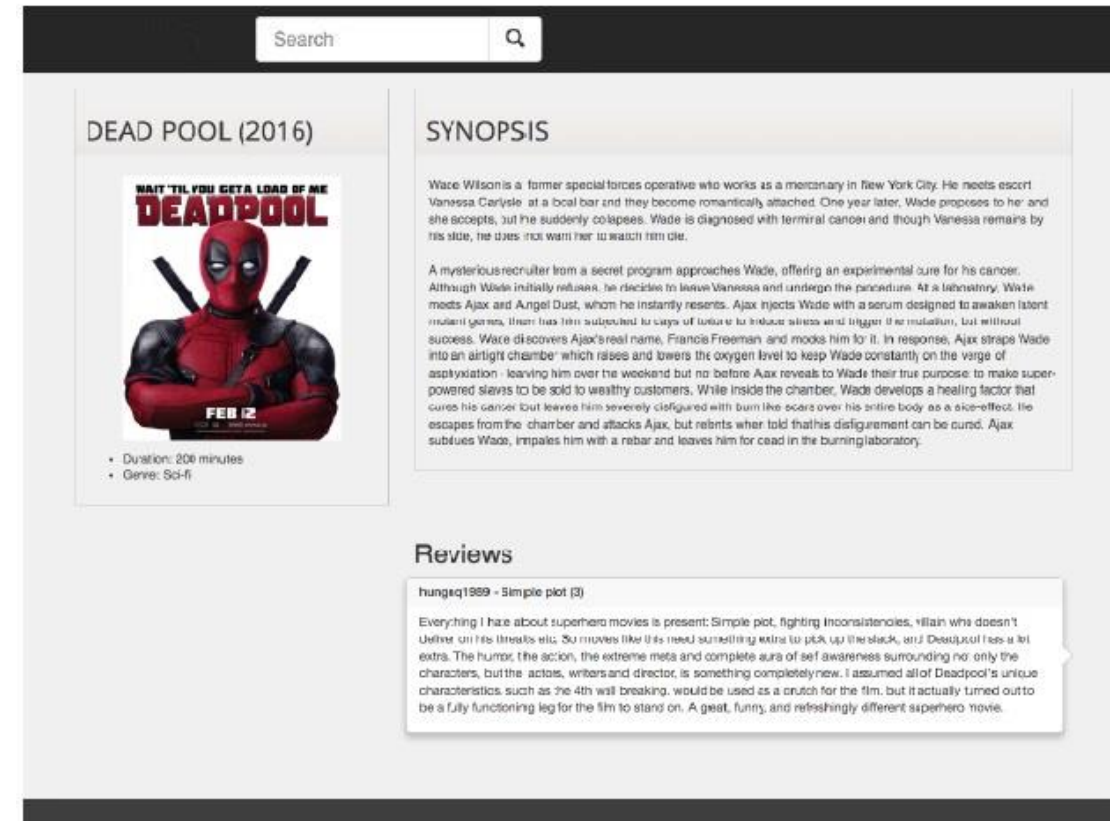
- List of all movies
 - each movie has a title, a thumbnail and a small description
 - the movie title is clickable and redirects to the details page
- Get request



EXERCISE 3- MOVIE APPLICATION

Details page

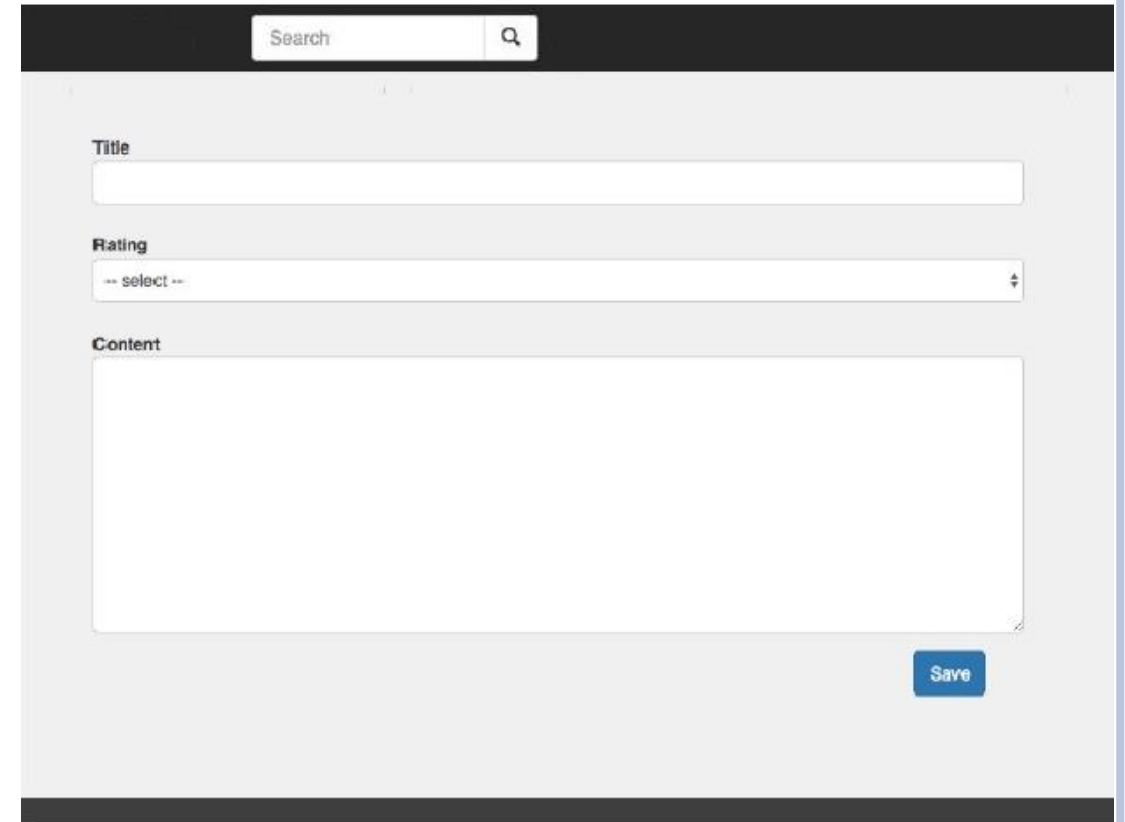
- It shows the :
 - movie title
 - production year
 - thumbnail
 - duration
 - genre
 - synopsis
 - list of reviews
 - each review has a title, user that wrote it, date and the description
- Get request with movie Id



EXERCISE 3- MOVIE APPLICATION

Add review page

- The page has a:
 - title
 - rating : 1 to 5
 - content
- Post request

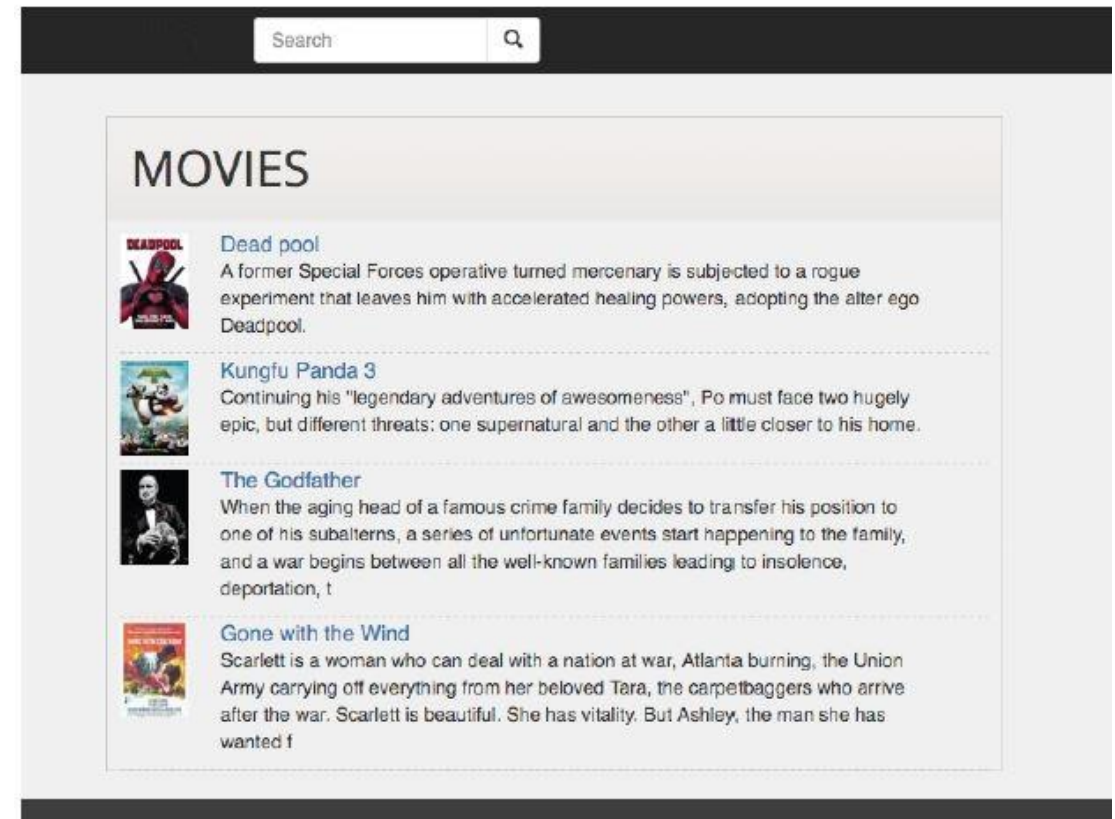


The screenshot shows a web form for adding a movie review. At the top, there is a dark header bar with a search input field labeled 'Search' and a magnifying glass icon. Below the header, the form is set against a light gray background. It contains three main sections: 'Title' with a text input field, 'Rating' with a dropdown menu showing '-- select --', and 'Content' with a large text area. A blue 'Save' button is located at the bottom right of the form.

EXERCISE 3- MOVIE APPLICATION

Search page

- User can by search movie title or description
- Results will be displayed as per the image here



EXERCISE 4- RESTFUL WEB API

The purpose of the exercise is to create a small restful API that will get, add, update and delete students.

In order, you need to create a new Laravel project and add the students table to the database.

You will also have to download and work with postman.

EXERCISE 4- RESTFUL WEB API

The API will have the following endpoints:

1. GET /api/students
 - will return all students and will be accepting GET requests.
2. GET /api/students/{id}
 - will return a student record by referencing its id and will be accepting GET requests.
 - Try to make the call for a non-existing id
3. POST /api/students
 - will create a new student record and will be accepting POST requests.
 - Call the get service to validate that a new student is created

EXERCISE 4- RESTFUL WEB API

The API will have the following endpoints:

4. PUT /api/students/{id}

- will update an existing student record by referencing its id and will be accepting PUT requests.
- Call the get service to validate that the record was updated

5. DELETE /api/students/{id}

- will delete a student record by referencing its id and will be accepting DELETE requests.
- Call the get service to validate that the record was deleted

EXERCISE 4- RESTFUL WEB API

General notes:

- Routes should be set in routes/api.php
- All responses should be of type json
- For Post and Put services, the request should be of type json
- Make sure to return the correct response code:
 - 200 ok
 - 404 not found
 - 500 internal sever error
 - etc.