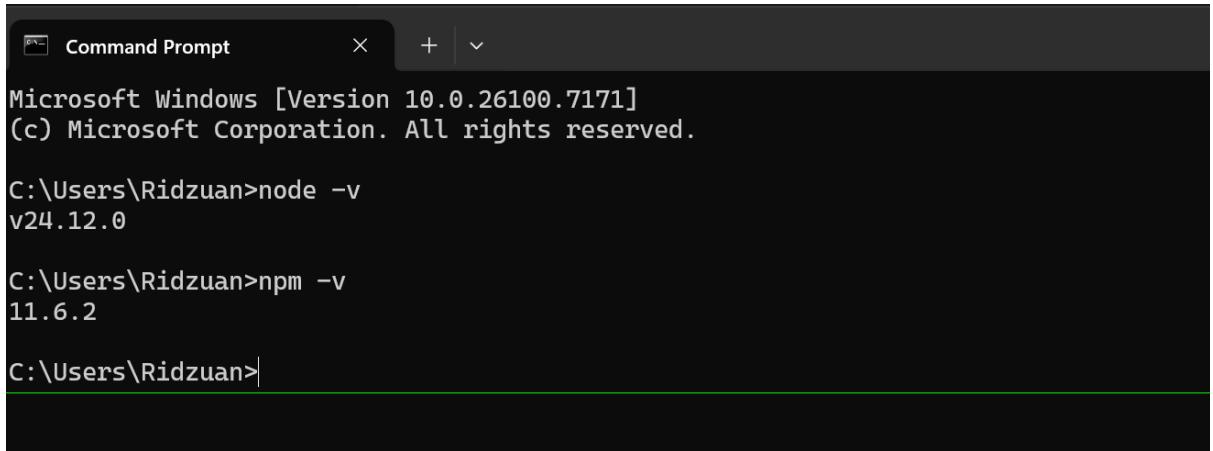


## TP MOD 14 WEBPRO

Muhammad Ilham Ridzuan – 103022300033

### 1. Download node.js



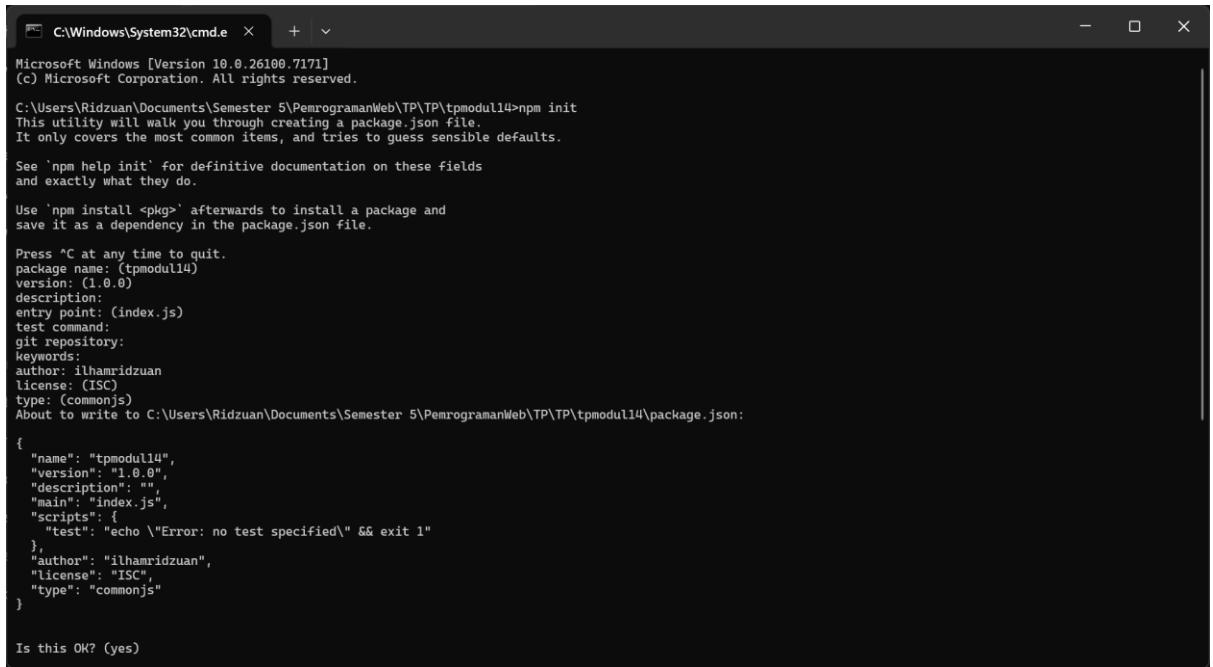
```
Command Prompt      + | v
Microsoft Windows [Version 10.0.26100.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ridzuan>node -v
v24.12.0

C:\Users\Ridzuan>npm -v
11.6.2

C:\Users\Ridzuan>
```

### 2.



```
C:\Windows\System32\cmd.e      + | v
Microsoft Windows [Version 10.0.26100.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ridzuan\Documents\Semester 5\PemrogramanWeb\TP\TP\tpmodul14>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
See 'npm help init' for definitive documentation on these fields
and exactly what they do.

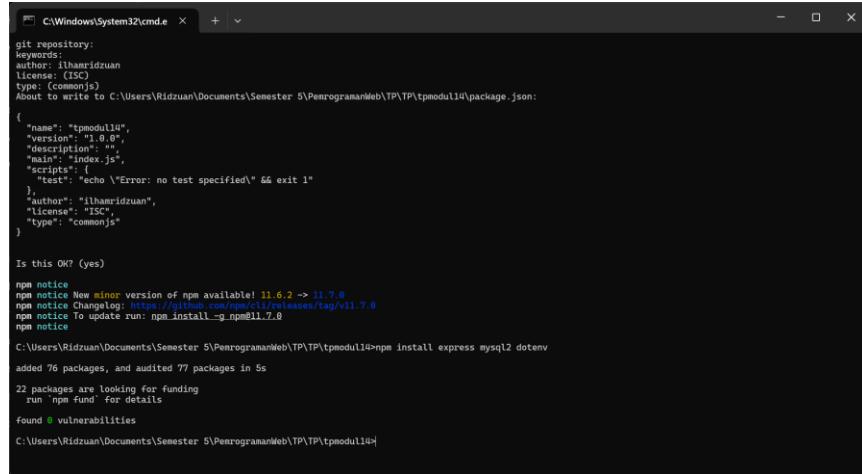
Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (tpmodul14)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: ilhamridzuan
license: (ISC)
type: (commonjs)
About to write to C:\Users\Ridzuan\Documents\Semester 5\PemrogramanWeb\TP\TP\tpmodul14\package.json:

{
  "name": "tpmodul14",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"$Error: no test specified\\" && exit 1"
  },
  "author": "ilhamridzuan",
  "license": "ISC",
  "type": "commonjs"
}

Is this OK? (yes)
```

### 3. Install express & dependency tambahan



```
C:\Windows\system32\cmd.exe + v
git repository:
keywords:
author: ihamridzuan
license: (ISC)
type: CommonJS
About to write to C:\Users\Ridzuan\Documents\Semester 5\PemrogramanWeb\TP\TP\tpmodul14\package.json:
{
  "name": "tpmodul14",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "ihamridzuan",
  "license": "ISC",
  "type": "commonjs"
}

Is this OK? (yes)

npm notice New minor version of npm available! 11.6.2 -> 11.7.0
npm notice ChangeLog: https://github.com/npm/cli/releases/tag/v11.7.0
npm notice Update run: npm install -g npm@11.7.0
npm notice

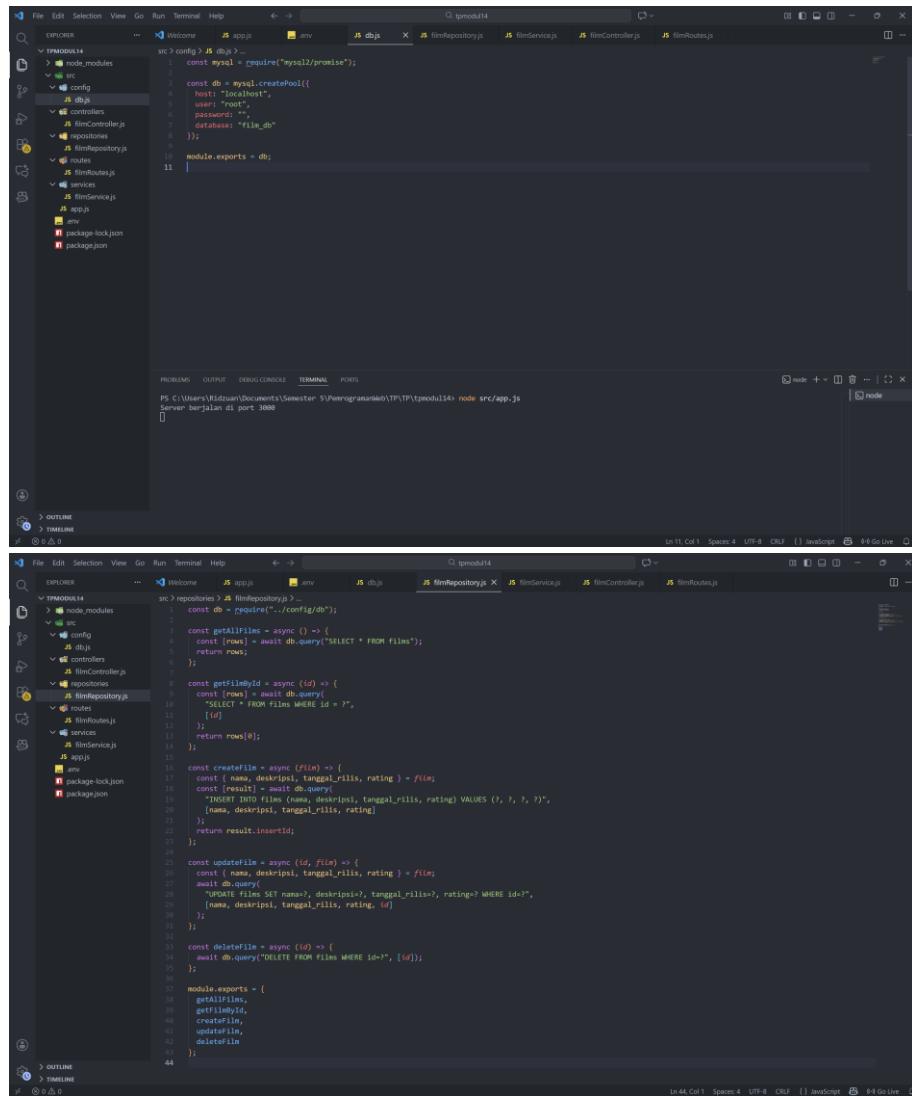
C:\Users\Ridzuan\Documents\Semester 5\PemrogramanWeb\TP\TP\tpmodul14>npm install express mysql2 dotenv
added 76 packages, and audited 77 packages in 5s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Ridzuan\Documents\Semester 5\PemrogramanWeb\TP\TP\tpmodul14
```

### 4. Project API FILM



The screenshot displays two Microsoft Visual Studio Code windows side-by-side, both showing the file structure and code for a Node.js project named 'tpmodul14'.

**File Structure:**

- Left Window (Terminal View):** Shows the file structure under 'tpmodul14'. The 'src' folder contains 'config', 'db.js', 'filmController.js', 'filmRepository.js', 'routes', 'services', and 'app.js'. Other files like 'env' and 'package-lock.json' are also present.
- Right Window (Code View):** Shows the content of 'db.js'. The code defines a MySQL pool with host 'localhost', user 'root', and database 'film\_db'. It then exports the pool as 'module.exports = db;'.

```
const mysql = require("mysql2/promise");
const db = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "",
  database: "film_db"
});
module.exports = db;
```

**File Content (filmRepository.js):**

```
const db = require("./config/db");
const getAllFilms = sync () => {
  const [rows] = await db.query("SELECT * FROM films");
  return rows;
};

const getFilmById = async (id) => {
  const [rows] = await db.query(`SELECT * FROM films WHERE id=?`, [id]);
  return rows[0];
};

const createFilm = async (film) => {
  const [name, deskripsi, tanggal_rilis, rating] = film;
  const result = await db.query(`INSERT INTO film (name, deskripsi, tanggal_rilis, rating) VALUES (?, ?, ?, ?)`, [name, deskripsi, tanggal_rilis, rating]);
  return result.insertId;
};

const updateFilm = async (id, film) => {
  const [name, deskripsi, tanggal_rilis, rating] = film;
  await db.query(`UPDATE films SET name=?, deskripsi=?, tanggal_rilis=?, rating=? WHERE id=?`, [name, deskripsi, tanggal_rilis, rating, id]);
};

const deleteFilm = async (id) => {
  await db.query("DELETE FROM films WHERE id=?", [id]);
};

module.exports = {
  getAllFilms,
  getFilmById,
  createFilm,
  updateFilm,
  deleteFilm
};
```

```
File Edit Selection View Go Run Terminal Help ⏎ → 🔍 tpmodule14
src > services > JS filmService.js
const filmRepository = require("./repositories/filmRepository");
...
const getAllFilms = async () => {
  return await filmRepository.getAllFilms();
}
...
const getFilmById = async (id) => {
  if (!id) {
    throw new Error("ID film wajib diisi");
  }
  return await filmRepository.getFilmById(id);
}
...
const createFilm = async (film) => {
  if (!film.name) {
    throw new Error("Nama film wajib diisi");
  }
  return await filmRepository.createFilm(film);
}
...
const updateFilm = async (id, film) => {
  if (!id) {
    throw new Error("ID film tidak valid");
  }
  await filmRepository.updateFilm(id, film);
}
...
const deleteFilm = async (id) => {
  if (!id) {
    throw new Error("ID film tidak valid");
  }
  await filmRepository.deleteFilm(id);
}
...
module.exports = {
  getAllFilms,
  getFilmById,
  createFilm,
  updateFilm,
  deleteFilm
};
```

Line 42, Col 1 | Spaces: 4 | UTF-8 | CR LF | {} | devLang | 88 Go Live

```
File Edit Selection View Go Run Terminal Help ⏎ → 🔍 tpmodule14
src > controllers > JS filmController.js
const filmService = require("../services/filmService");
...
const getAllFilms = async (req, res) => {
  try {
    const films = await filmService.getAllFilms();
    res.status(200).json(films);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
...
const getFilmById = async (req, res) => {
  try {
    const film = await filmService.getFilmById(req.params.id);
    res.status(200).json(film);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
...
const createFilm = async (req, res) => {
  try {
    const id = await filmService.createFilm(req.body);
    res.status(201).json({ message: "Film berhasil ditambahkan", id });
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
...
const updateFilm = async (req, res) => {
  try {
    await filmService.updateFilm(req.params.id, req.body);
    res.status(200).json({ message: "Film berhasil diperbarui" });
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
...
const deleteFilm = async (req, res) => {
  try {
    await filmService.deleteFilm(req.params.id);
    res.status(200).json({ message: "Film berhasil dihapus" });
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
...
module.exports = {
  getAllFilms,
  getFilmById,
  createFilm,
  updateFilm,
  deleteFilm
};
```

Line 55, Col 1 | Spaces: 4 | UTF-8 | CR LF | {} | devLang | 84 Go Live

```
File Edit Selection View Go Run Terminal Help ⏎ → 🔍 tpmodule14
src > routes > JS filmRoutes.js
const express = require("express");
const router = express.Router();
const filmController = require("../controllers/filmController");
...
router.get("/", filmController.getAllFilms);
router.get("/:id", filmController.getFilmById);
router.post("/", filmController.createFilm);
router.put("/:id", filmController.updateFilm);
router.delete("/:id", filmController.deleteFilm);
...
module.exports = router;
```

Line 12, Col 1 | Spaces: 4 | UTF-8 | CR LF | {} | devLang | 84 Go Live

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'TMODUL14' containing several files: 'node\_modules', 'config', 'controllers', 'routes', 'services', and 'app.js'. The 'app.js' file is open in the editor, displaying the following code:

```

src > 26 app.js
  const express = require('express');
  const app = express();

  app.use(express.json());
  app.use('/film', require('./routes/filmRoutes'));

  app.listen(3000, () => {
    console.log('Server berjalan di port 3000');
  });

```

In the terminal at the bottom, the command 'node src/app.js' is run, resulting in the output 'Server berjalan di port 3000'.

## TESTING API

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections, environments, history, and flows. The main workspace shows a 'Get data' collection with a 'POST Get data' request. The request details are as follows:

- Method: POST
- URL: http://localhost:3000/films/
- Body type: raw (JSON)
- Body content:

```

1 [
2   {
3     "nama": "MAI",
4     "deskripsi": "Film Netflix dari vietnam",
5     "tanggal_rilis": "2024-03-01",
6     "rating": 9.0
7   }
8 ]

```

The response section shows a 201 Created status with the message: "Film berhasil ditambahkan", and an id of 1.

The screenshot shows the Postman application interface again. The 'My Workspace' sidebar is visible. The main workspace shows a 'Get data' collection with a 'GET Get data' request. The request details are as follows:

- Method: GET
- URL: http://localhost:3000/films/
- Body type: raw (JSON)
- Body content:

The response section shows a 200 OK status with the following JSON data:

```

1 [
2   {
3     "id": 1,
4     "nama": "MAI",
5     "deskripsi": "Film Netflix dari vietnam",
6     "tanggal_rilis": "2024-02-29T17:00:00.000Z",
7     "rating": 9
8   },
9   {
10    "id": 2,
11    "nama": "Man In Love",
12    "deskripsi": "film Diana Romance Taiwan",
13    "tanggal_rilis": "2021-08-19T17:00:00.000Z",
14    "rating": 9.5
15  }
16 ]

```

The image consists of three vertically stacked screenshots of the Postman application interface, demonstrating the use of different HTTP methods on a REST API endpoint.

**Top Screenshot (GET Request):**

- Method:** GET
- URL:** `http://localhost:3000/films/2`
- Body:** JSON (Raw) - Displays the response body as JSON, representing a film record with fields: id, nama, deskripsi, tanggal\_rilis, and rating.
- Response Headers:** 200 OK, 8 ms, 361 B

```
1 {
2   "id": 2,
3   "nama": "Man In Love",
4   "deskripsi": "Film Drama Romance Taiwan yang tayang di netflix",
5   "tanggal_rilis": "2021-08-19T17:00:00Z",
6   "rating": 9.5
7 }
```

**Middle Screenshot (PUT Request):**

- Method:** PUT
- URL:** `http://localhost:3000/films/2`
- Body:** JSON (Raw) - Displays the updated film record with the name changed to "Man In Love".
- Response Headers:** 200 OK, 7 ms, 273 B

```
1 {
2   "nama": "Man In Love",
3   "deskripsi": "Film Drama Romance Taiwan yang tayang di netflix",
4   "tanggal_rilis": "2021-08-20",
5   "rating": 9.5
6 }
```

**Bottom Screenshot (DELETE Request):**

- Method:** DELETE
- URL:** `http://localhost:3000/films/2`
- Body:** JSON (Raw) - Displays a message indicating the film was successfully deleted.
- Response Headers:** 200 OK, 10 ms, 270 B

```
1 {
2   "message": "Film berhasil diperbarui"
3 }
```

## Project Laravel

The image displays three screenshots of a Laravel application interface, showing the 'Daftar Film' (List of Films) page, the 'Tambah Film' (Add Film) page, and the 'Edit Film' (Edit Film) page.

**Daftar Film (List of Films) Page:**

Nama	Rating	Aksi
MAI	9	<a href="#">Edit</a> <a href="#">Hapus</a>
Man In Love	9	<a href="#">Edit</a> <a href="#">Hapus</a>

**Tambah Film (Add Film) Page:**

**Fields:**

- Nama: A Man Called Otto
- Deskripsi: Film Comedy-Drama dari Amerika
- Tanggal Rilis: 01/13/2023
- Rating: 8

**Edit Film (Edit Film) Page:**

**Fields:**

- Nama: MAI
- Deskripsi: Film Netflix dari vietnam
- Tanggal Rilis: mm/dd/yyyy
- Rating: 8.5

Nama	Rating	Aksi
AI	8.5	<button>Edit</button> <button>Hapus</button>
Man In Love	9	<button>Edit</button> <button>Hapus</button>
Frankenstein	9.5	<button>Edit</button> <button>Hapus</button>
A Man Called Otto	8	<button>Edit</button> <button>Hapus</button>

Setelah hapus film ke-4

Nama	Rating	Aksi
AI	8.5	<button>Edit</button> <button>Hapus</button>
Man In Love	9	<button>Edit</button> <button>Hapus</button>
Frankenstein	9.5	<button>Edit</button> <button>Hapus</button>

Code Laravel :  
FilmController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;

class FilmController extends Controller
{
    private $apiUrl = 'http://localhost:3000/films';

    public function index()
    {
```

```

        $response = Http::get($this->apiUrl);
        $films = $response->json();
        return view('index', compact('films'));
    }

    public function create()
    {
        return view('create');
    }

    public function store(Request $request)
    {
        Http::post($this->apiUrl, $request->all());
        return redirect('/films');
    }

    public function edit($id)
    {
        $film = Http::get($this->apiUrl.'/'.$id)->json();
        return view('edit', compact('film'));
    }

    public function update(Request $request, $id)
    {
        Http::put($this->apiUrl.'/'.$id, $request->all());
        return redirect('/films');
    }

    public function destroy($id)
    {
        Http::delete($this->apiUrl.'/'.$id);
        return redirect('/films');
    }
}

```

## Routes

```

<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\FilmController;

Route::get('/films', [FilmController::class, 'index']);
Route::get('/films/create', [FilmController::class, 'create']);
Route::post('/films', [FilmController::class, 'store']);
Route::get('/films/{id}/edit', [FilmController::class, 'edit']);
Route::put('/films/{id}', [FilmController::class, 'update']);
Route::delete('/films/{id}', [FilmController::class, 'destroy']);

```

## App.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Film App</title>
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
        rel="stylesheet">
</head>
<body>
<div class="container mt-4">
    @yield('content')
</div>
</body>
</html>
```

## index.blade.php

```
@extends('app')

@section('content')
<h2>Daftar Film</h2>

<a href="/films/create" class="btn btn-primary mb-3">Tambah Film</a>

<table class="table table-bordered">
    <tr>
        <th>Nama</th>
        <th>Rating</th>
        <th>Aksi</th>
    </tr>
    @foreach ($films as $film)
    <tr>
        <td>{{ $film['nama'] }}</td>
        <td>{{ $film['rating'] }}</td>
        <td>
            <a href="/films/{{ $film['id'] }}/edit" class="btn btn-warning btn-sm">Edit</a>
            <form action="/films/{{ $film['id'] }}" method="POST" style="display:inline;">
                @csrf
                @method('DELETE')
                <button class="btn btn-danger btn-sm">Hapus</button>
            </form>
        </td>
    </tr>
    @endforeach
</table>
```

```
</table>
@endsection
```

### create.blade.php

```
@extends('app')

@section('content')
<h2>Tambah Film</h2>

<form method="POST" action="/films">
@csrf
<div class="mb-2">
    <label>Nama</label>
    <input type="text" name="nama" class="form-control">
</div>

<div class="mb-2">
    <label>Deskripsi</label>
    <textarea name="deskripsi" class="form-control"></textarea>
</div>

<div class="mb-2">
    <label>Tanggal Rilis</label>
    <input type="date" name="tanggal_rilis" class="form-control">
</div>

<div class="mb-2">
    <label>Rating</label>
    <input type="number" step="0.1" name="rating" class="form-control">
</div>

<button class="btn btn-success">Simpan</button>
</form>
@endsection
```

### edit.blade.php

```
@extends('app')

@section('content')
<h2>Edit Film</h2>

<form method="POST" action="/films/{{ $film['id'] }}">
@csrf
@method('PUT')
```

```
<div class="mb-2">
    <label>Nama</label>
    <input type="text" name="nama" value="{{ $film['nama'] }}" class="form-control">
</div>

<div class="mb-2">
    <label>Deskripsi</label>
    <textarea name="deskripsi" class="form-control">{{ $film['deskripsi'] }}</textarea>
</div>

<div class="mb-2">
    <label>Tanggal Rilis</label>
    <input type="date" name="tanggal_rilis" value="{{ $film['tanggal_rilis'] }}" class="form-control">
</div>

<div class="mb-2">
    <label>Rating</label>
    <input type="number" step="0.1" name="rating" value="{{ $film['rating'] }}" class="form-control">
</div>

<button class="btn btn-primary">Update</button>
</form>
@endsection
```