

LAPORAN PRAKTIKUM PEMROGRAMAN WEB
PRAKTIKUM 14 – CREATE READ UPDATE & DELETE



Disusun oleh:

Nama : Alfath Roziq Widhayaka

Nim : L0122012

Kelas : A

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET

2024

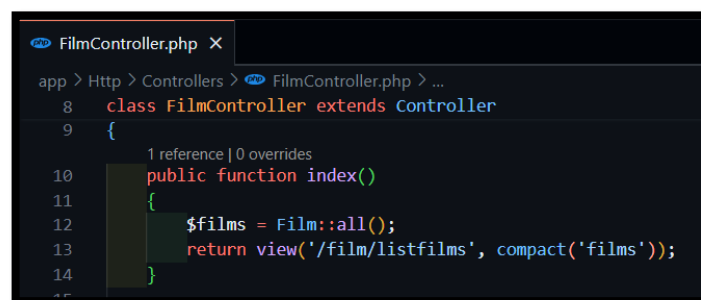
BAB I

A. Penjelasan Source Code

Pada praktikum kali ini, saya melanjutkan praktikum sebelumnya dengan menambahkan fungsi **CRUD (Create, Read, Update, dan Delete)** yang dinamis pada salah satu database yang saya miliki. Kegunaan dari implementasi CRUD ini adalah untuk memungkinkan interaksi yang lebih efektif dan efisien dengan data dalam database. Dengan fungsi **Create**, kita dapat menambahkan data baru ke dalam database. Fungsi **Read** memungkinkan kita untuk membaca dan menampilkan data yang ada. Melalui fungsi **Update**, kita dapat memperbarui data yang sudah ada sesuai kebutuhan. Terakhir, fungsi **Delete** digunakan untuk menghapus data yang tidak diperlukan lagi. Dengan adanya keempat fungsi ini, manajemen data menjadi lebih fleksibel dan dapat disesuaikan dengan perubahan kebutuhan, memastikan integritas dan kemitakhiran data selalu terjaga. Berikut merupakan implementasi CRUD pada laravel milik saya.

1. Melengkapi Controller

- **Index Function**

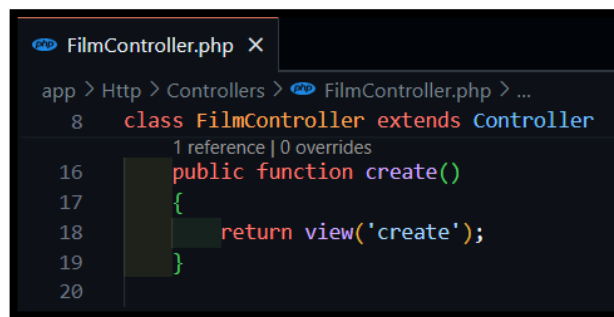


```
FilmController.php X
app > Http > Controllers > FilmController.php > ...
8  class FilmController extends Controller
9  {
10     1 reference | 0 overrides
11     public function index()
12     {
13         $films = Film::all();
14         return view('/film/listfilms', compact('films'));
15     }
16 }
```

Gambar 1 index function

Fungsi **index** bertanggung jawab untuk menampilkan semua data film yang tersimpan dalam database. Pertama, fungsi ini menggunakan **Film::all()** untuk mengambil semua record film dari tabel films dan menyimpannya dalam variabel **\$films**. Setelah itu, data ini dikirimkan ke view **film/listfilms** menggunakan fungsi **compact** yang membuat array asosiatif dari variabel yang diberikan. View tersebut kemudian akan menampilkan daftar film kepada pengguna. Fungsi ini sangat penting untuk menyediakan tampilan utama dari semua film yang ada.

- **Create Function**

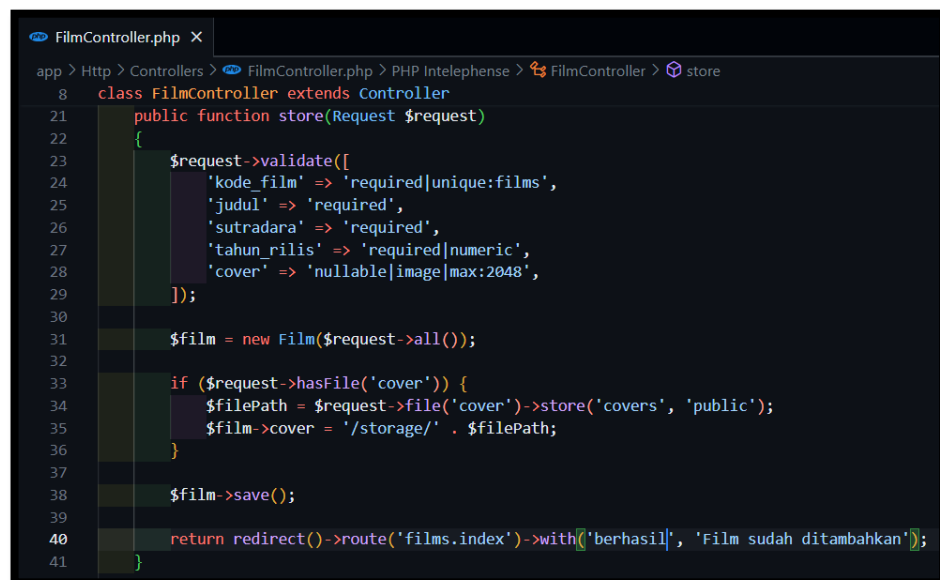


```
FilmController.php X
app > Http > Controllers > FilmController.php > ...
8 class FilmController extends Controller
  1 reference | 0 overrides
16 public function create()
17 {
18     return view('create');
19 }
20
```

Gambar 2 create function

Fungsi **create** berfungsi untuk menampilkan form input yang digunakan untuk membuat film baru. Fungsi ini hanya merender view **create** yang berisi form yang diperlukan untuk mengisi data seperti kode film, judul, sutradara, tahun rilis, dan cover film. Fungsi ini tidak memerlukan data dari database dan hanya berfokus pada penyediaan form input bagi pengguna.

- **Store Function**



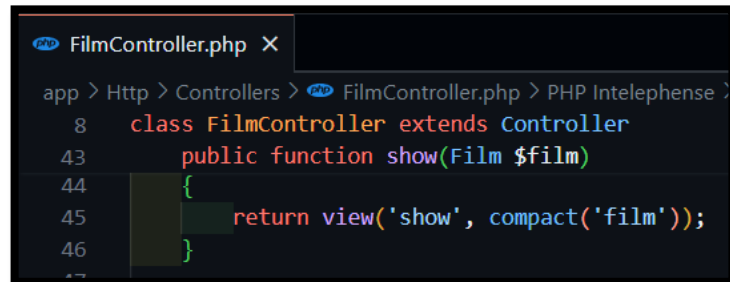
```
FilmController.php X
app > Http > Controllers > FilmController.php > PHP Intelephense > FilmController > store
8 class FilmController extends Controller
21 public function store(Request $request)
22 {
23     $request->validate([
24         'kode_film' => 'required|unique:films',
25         'judul' => 'required',
26         'sutradara' => 'required',
27         'tahun_rilis' => 'required|numeric',
28         'cover' => 'nullable|image|max:2048',
29     ]);
30
31     $film = new Film($request->all());
32
33     if ($request->hasFile('cover')) {
34         $filePath = $request->file('cover')->store('covers', 'public');
35         $film->cover = '/storage/' . $filePath;
36     }
37
38     $film->save();
39
40     return redirect()->route('films.index')->with(['berhasil' => 'Film sudah ditambahkan']);
41 }
```

Gambar 3 store function

Fungsi **store** bertugas untuk menyimpan data film baru ke dalam database setelah form input di-submit oleh pengguna. Pertama, fungsi ini melakukan validasi terhadap data yang diterima dari request menggunakan metode validate. Data yang harus diisi termasuk **kode_film**, **judul**, **sutradara**, dan **tahun_rilis**, sedangkan **cover** bersifat opsional dan harus berupa file gambar dengan ukuran maksimum 2048 KB. Jika terdapat file cover, file tersebut akan disimpan di direktori

public/covers dan path-nya disimpan dalam atribut cover pada model Film. Kemudian, objek Film yang baru diisi dengan data dari request disimpan ke database. Setelah proses penyimpanan selesai, pengguna akan diarahkan kembali ke halaman daftar film dengan pesan sukses yang menunjukkan bahwa film baru telah berhasil ditambahkan.

- **Show Function**

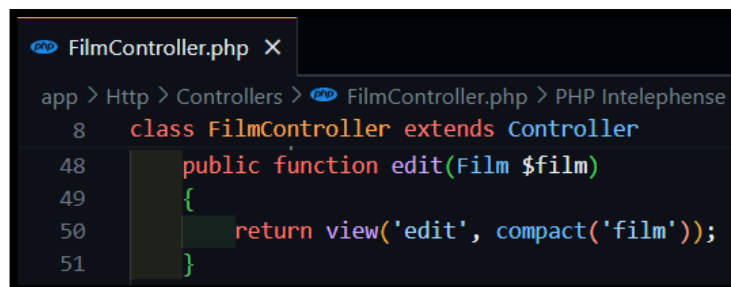


```
FilmController.php X
app > Http > Controllers > FilmController.php > PHP Intelephense
8  class FilmController extends Controller
43  public function show(Film $film)
44  {
45      return view('show', compact('film'));
46  }
```

Gambar 4 show function

Fungsi **show** digunakan untuk menampilkan detail lengkap dari satu film tertentu. Fungsi ini menerima objek Film yang diambil berdasarkan ID yang diberikan di URL. Data film tersebut kemudian dikirimkan ke view show menggunakan fungsi compact, yang akan menampilkan detail film seperti judul, sutradara, tahun rilis, dan cover kepada pengguna. Fungsi ini berguna untuk memberikan informasi lebih mendetail tentang satu film tertentu kepada pengguna.

- **Edit Function**



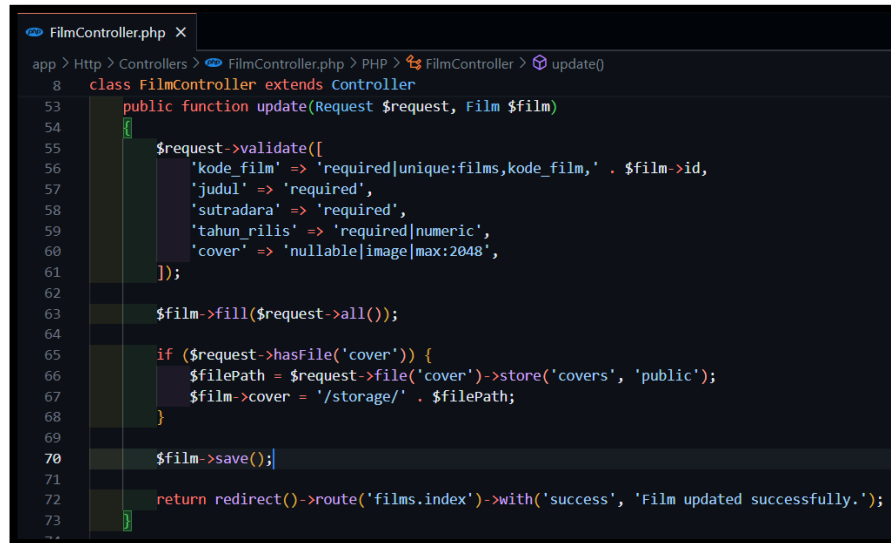
```
FilmController.php X
app > Http > Controllers > FilmController.php > PHP Intelephense
8  class FilmController extends Controller
48  public function edit(Film $film)
49  {
50      return view('edit', compact('film'));
51  }
```

Gambar 5 edit function

Fungsi **edit** digunakan untuk menampilkan form edit yang sudah terisi dengan data film tertentu yang ingin diubah. Fungsi ini menerima objek Film yang diambil berdasarkan ID yang diberikan di URL dan mengirimkannya ke view **edit**. View ini akan menampilkan form dengan data yang sudah ada, sehingga pengguna dapat

melihat dan mengubah data yang diperlukan. Setelah pengguna selesai mengedit, data tersebut dapat disubmit untuk diperbarui dalam database.

- **Update Function**

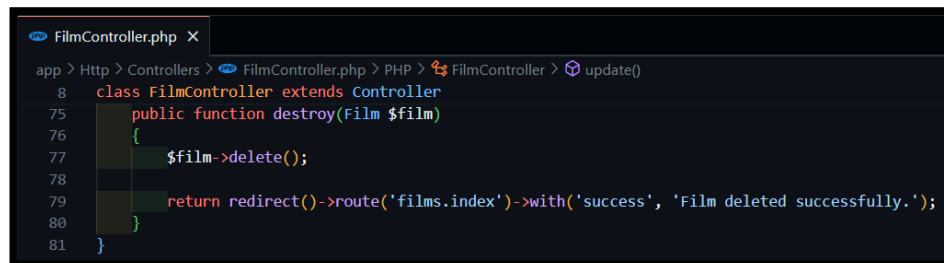


```
app > Http > Controllers > FilmController.php > PHP > FilmController > update()
8 class FilmController extends Controller
53 public function update(Request $request, Film $film)
54 {
55     $request->validate([
56         'kode_film' => 'required|unique:films,kode_film,' . $film->id,
57         'judul' => 'required',
58         'sutradara' => 'required',
59         'tahun_rilis' => 'required|numeric',
60         'cover' => 'nullable|image|max:2048',
61     ]);
62
63     $film->fill($request->all());
64
65     if ($request->hasFile('cover')) {
66         $filePath = $request->file('cover')->store('covers', 'public');
67         $film->cover = '/storage/' . $filePath;
68     }
69
70     $film->save();
71
72     return redirect()->route('films.index')->with('success', 'Film updated successfully.');
```

Gambar 6 update function

Fungsi **update** bertanggung jawab untuk memperbarui data film yang sudah ada di database dengan data baru yang diterima dari form edit. Fungsi ini pertama-tama melakukan validasi terhadap data yang diterima dari request, memastikan bahwa **kode_film**, **judul**, **sutradara**, dan **tahun_rilis** valid. Selain itu, validasi memastikan bahwa **kode_film** unik kecuali untuk film yang sedang diupdate. Jika ada file cover baru yang diupload, file tersebut akan disimpan di direktori **public/covers** dan path-nya diperbarui dalam atribut **cover** dari objek **Film**. Setelah semua data diupdate, objek **Film** disimpan kembali ke database. Fungsi ini kemudian mengarahkan pengguna kembali ke halaman daftar film dengan pesan sukses yang menunjukkan bahwa film telah berhasil diperbarui.

▪ Destroy Function

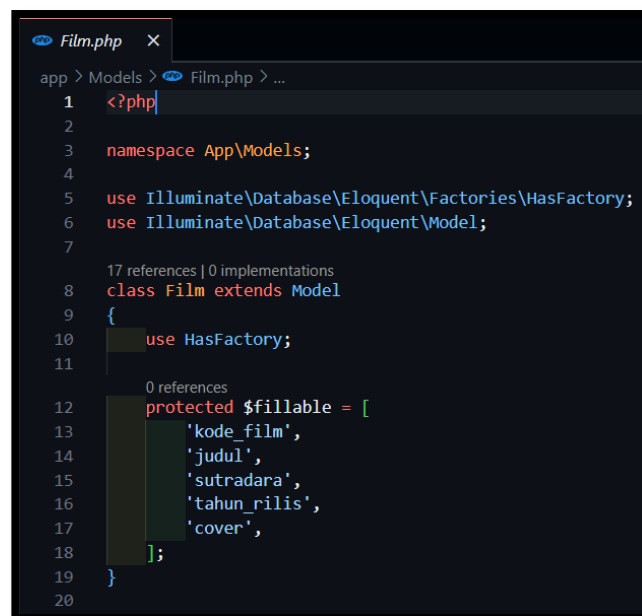


```
app > Http > Controllers > FilmController.php > PHP > FilmController > update()
8  class FilmController extends Controller
75 public function destroy(Film $film)
76 {
77     $film->delete();
78
79     return redirect()->route('films.index')->with('success', 'Film deleted successfully.');
```

Gambar 7 destroy function

Fungsi **destroy** bertanggung jawab untuk menghapus data film tertentu dari database. Fungsi ini menerima objek Film yang diambil berdasarkan ID yang diberikan di URL dan menghapusnya dari database menggunakan metode delete. Setelah penghapusan berhasil, pengguna diarahkan kembali ke halaman daftar film dengan pesan sukses yang menunjukkan bahwa film telah berhasil dihapus. Fungsi ini penting untuk memungkinkan pengguna mengelola dan menghapus film yang tidak diperlukan lagi.

2. Melengkapi Model



```
app > Models > Film.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Film extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'kode_film',
14         'judul',
15         'sutradara',
16         'tahun_rilis',
17         'cover',
18     ];
19 }
20
```

Gambar 8 model Film.php

Model Film diatas merupakan representasi dalam kode dari entitas film dalam aplikasi Laravel. Dengan mendefinisikan atribut yang dapat diisi secara massal seperti **kode_film**, **judul**, **sutradara**, **tahun_rilis**, dan **cover** dalam array **\$fillable**, model ini memungkinkan penggunaan metode seperti **create** dan **update** dengan aman.

3. Melengkapi Route



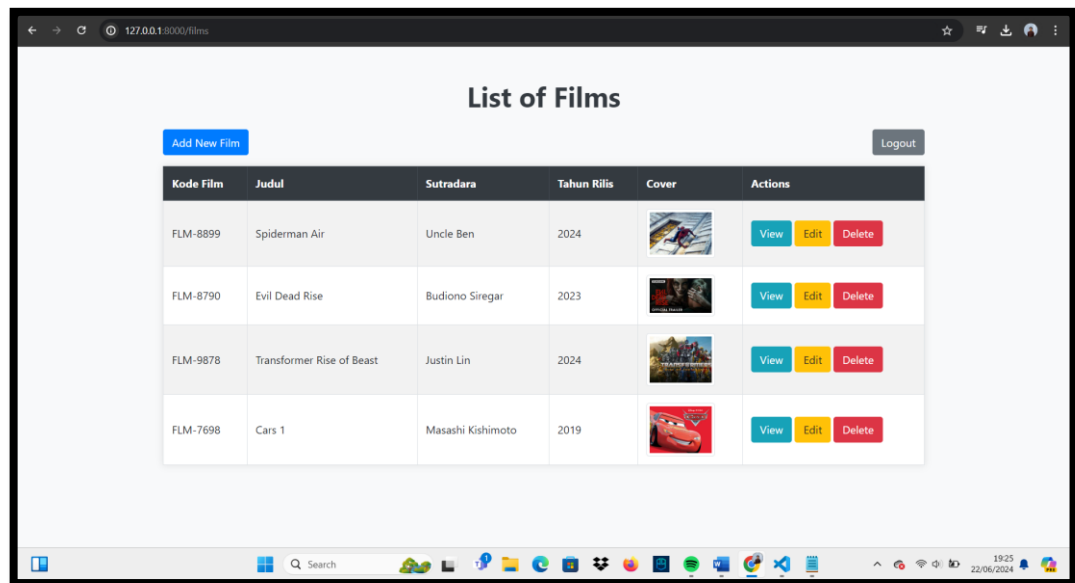
Gambar 9 web.php

File web.php diatas adalah definisi **route-resource untuk entitas films**, dimana setiap route (**index, create, store, show, edit, update, destroy**) terhubung ke method-method yang sesuai dalam **FilmController**. Route ini memberikan endpoint untuk menampilkan daftar film, menambahkan film baru, menyimpan film baru, menampilkan detail film, mengedit film, memperbarui data film, dan menghapus film dari database, memudahkan manajemen **CRUD (Create, Read, Update, Delete)** data film dalam aplikasi Laravel secara efisien dan terstruktur.

BAB II

A. Hasil Output

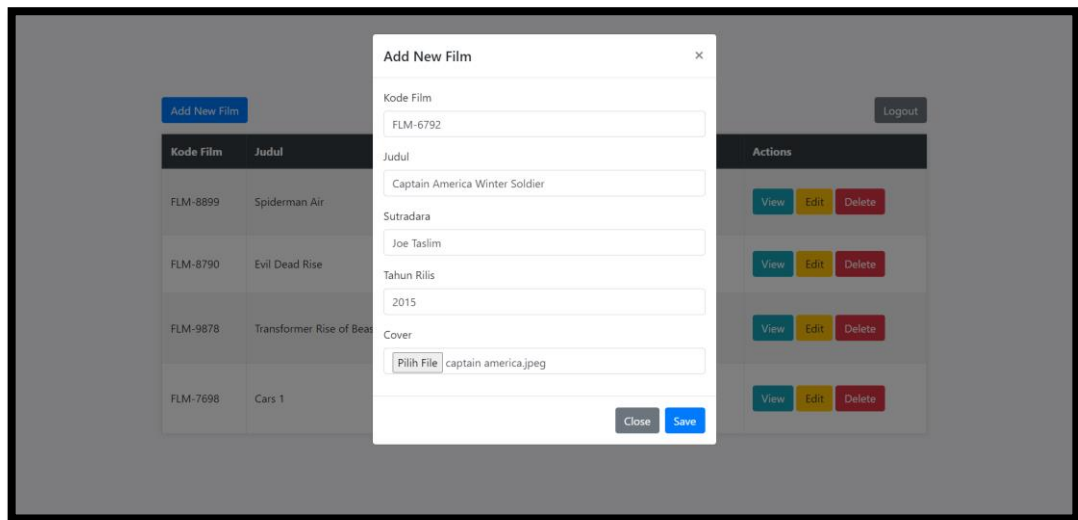
- Index



Gambar 10 index

Gambar diatas merupakan tampilan ketika user berada di halaman index. Pada halaman index ini menampilkan daftar film yang telah di tambahkan oleh saya sendiri. Terdapat tombol **Add New Film** untuk menambahkan film, tombol **View** untuk melihat detail dari salah satu film pada tabel tersebut, toombol **Edit** untuk mengedit detail isi film, dan terakhir tombol **Delete** untuk menghapus salah satu film.

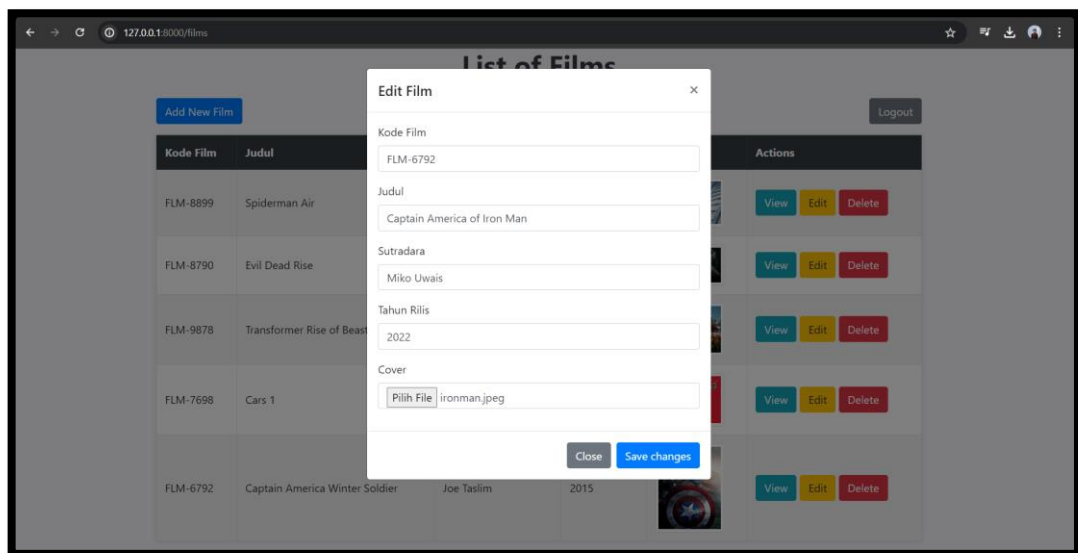
- **Create**



Gambar 11 create

Gambar diatas merupakan tampilan ketika pengguna ingin menambahkan film yang akan didaftarkan pada list film. Terdapat isian yang harus diisi oleh user seperti kode film, judul, sutradara, tahun rilis, dan cover film.

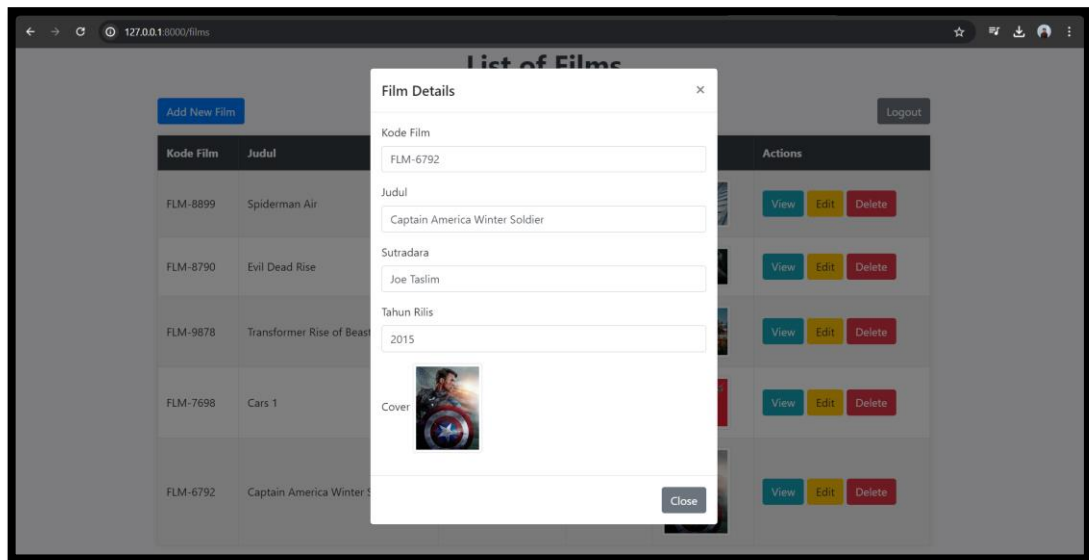
- **Update**



Gambar 12 update

Gambar diatas merupakan tampilan ketika pengguna ingin mengupdate film yang ingin diedit isi detail filmnya. Sama seperti menambahkan film yaitu terdapat isian yang harus diisi oleh user seperti kode film, judul, sutradara, tahun rilis, dan cover film.

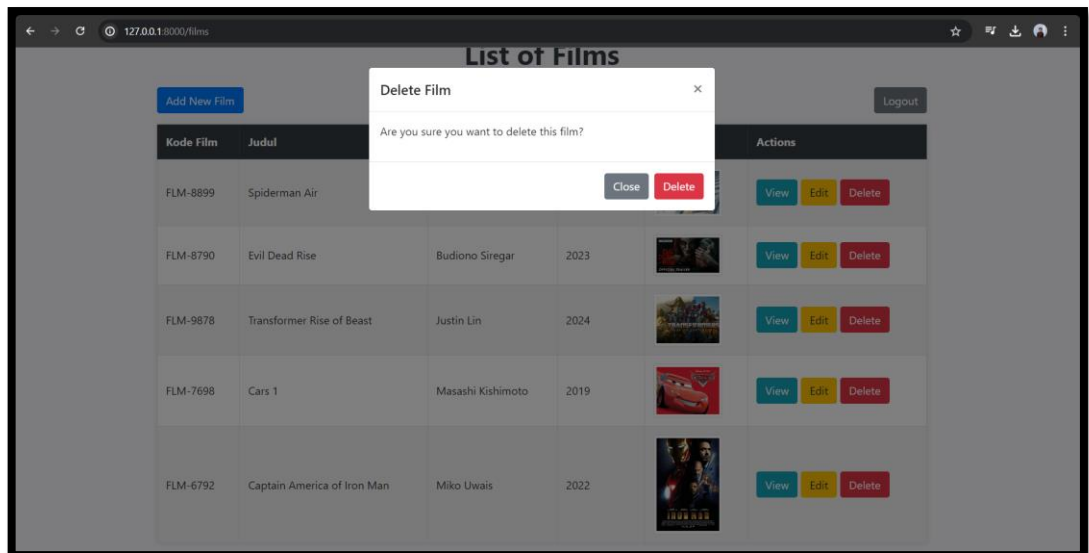
- **Show**



Gambar 13 show

Gambar diatas merupakan tampilan ketika pengguna ingin melihat isi detail pada salah satu film yang berada di daftar film.

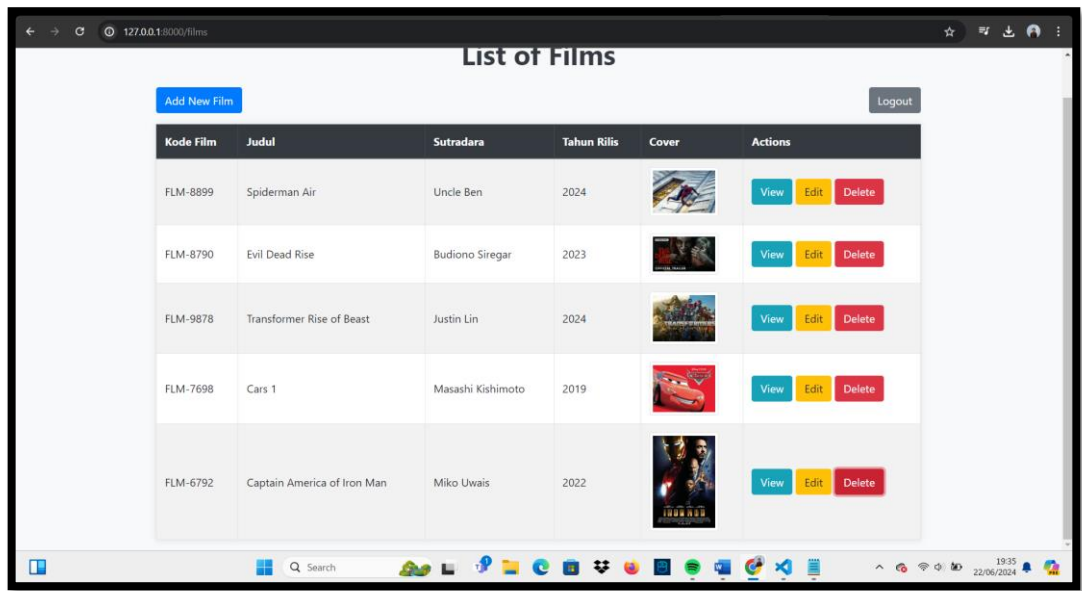
- **Delete**



Gambar 14 delete

Gambar diatas merupakan tampilan ketika pengguna ingin menghapus salah satu data film pada tabel. Terlihat terdapat peringatan ketika user ingin menghapus untuk memastikan user ingin menghapus film atau tidak.

- Hasil setelah ditambahkan



Gambar 15 hasil setelah ditambah

Gambar diatas merupakan tampilan ketika pengguna sudah menambahkan film, terlihat film **Captain America of Iron Man** sudah berhasil ditambahkan.

BAB III

A. Kesimpulan

Pada praktikum ini, saya berhasil mengimplementasikan fungsi **CRUD (Create, Read, Update, Delete)** pada aplikasi web menggunakan framework Laravel. CRUD memungkinkan saya untuk secara efektif mengelola data **film** dalam database, mulai dari menampilkan daftar film, menambahkan data baru, mengedit detail film, hingga menghapus entri yang tidak diperlukan. Dengan menggunakan controller yang telah diperluas dengan fungsi-fungsi seperti **index** untuk menampilkan semua film, **create** untuk menambahkan film baru, **show** untuk melihat detail film, **edit** untuk mengubah data film, dan **destroy** untuk menghapus film, aplikasi ini menjadi lebih dinamis dan responsif terhadap interaksi pengguna. Melalui penggunaan model Film yang terdefinisi dengan baik, serta konfigurasi route-resource yang memetakan setiap aksi CRUD ke method-method dalam **FilmController**.