

## Dokumentasi Code

Dalam metode aplikasi ini menggunakan metode CVM, atau Controller-View-Model, adalah singkatan yang merujuk pada pola arsitektur pengembangan perangkat lunak yang umumnya dikenal sebagai Model-View-Controller (MVC). Dalam konteks ini, setiap komponen memiliki peran spesifik:

### 1. Controller (C):

- Bertanggung jawab untuk menangani input dari pengguna, memprosesnya, dan memperbarui model dan/atau tampilan sesuai kebutuhan.
- Di dalam pengembangan web, controller menerima input dari pengguna, memprosesnya, dan berinteraksi dengan model untuk memperbarui data. Selanjutnya, data yang diperbarui diteruskan ke tampilan untuk ditampilkan.

### 2. View (V):

- Bertanggung jawab untuk menampilkan data kepada pengguna dan menerima input dari pengguna. Tampilan menampilkan informasi yang diperoleh dari model dan mengirim input pengguna kembali ke controller untuk diproses.
- Dalam pengembangan web, tampilan sering terkait dengan antarmuka pengguna (UI) atau representasi HTML/CSS dari aplikasi. Tampilan tidak menangani logika bisnis tetapi berfokus pada cara data ditampilkan dengan ramah pengguna.

### 3. Model (M):

- Mewakili data dan logika bisnis aplikasi. Model mengelola data, merespons permintaan informasi, dan memperbarui dirinya sendiri (atau memberi tahu pengamat) ketika terjadi perubahan.
- Dalam pengembangan web, model berinteraksi dengan basis data atau sumber data lain untuk mengambil dan memanipulasi data. Ini memberikan lapisan abstraksi untuk menangani tugas terkait data.

Dalam CVM, kita mengikuti pola MVC dengan pemisahan tanggung jawab masing-masing komponen:

- Controller: Menangani input pengguna, memprosesnya, dan berinteraksi dengan model dan tampilan.
- View: Menampilkan data kepada pengguna dan menangani input pengguna, yang diteruskan ke controller.
- Model: Mengelola data dan logika bisnis, berinteraksi dengan basis data atau sumber data lainnya.

Pola ini membantu dalam mengorganisir kode secara terstruktur, memisahkan peran-peran yang berbeda, dan memudahkan perawatan dan pengembangan aplikasi.

Table yang digunakan dalam PHPmyadmin (database)

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
<input type="checkbox"/> anggota	Jelajahi  Struktur  Cari  Tambahkan  Kosongkan  Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
<input type="checkbox"/> katalog	Jelajahi  Struktur  Cari  Tambahkan  Kosongkan  Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
<input type="checkbox"/> migrations	Jelajahi  Struktur  Cari  Tambahkan  Kosongkan  Hapus	8	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
<input type="checkbox"/> peminjaman	Jelajahi  Struktur  Cari  Tambahkan  Kosongkan  Hapus	0	InnoDB	utf8mb4_unicode_ci	48.0 KB	-

Gambar 1

Disini karena saya menggunakan 3 CRUD yaitu CRUD anggota, CRUD katalog dan juga CRUD peminjaman sehingga saya menggunakan 3 table yang mewakili setiap CRUD

1. Mein Menu

# Aplikasi Perpustakaan



Gambar 2

Pada Main menu awal terdapat 3 button yang akan membawa user menuju 3 fungsi utama yaitu CRUD anggota, CRUD katalog dan juga CRUD peminjaman.

2. Anggota

				id	nama	alamat	no_telepon	created_at	updated_at
<input type="checkbox"/>	Ubah	Salin	Hapus	1	Nico	Mojokerto	091	2024-01-06 20:27:27	2024-01-06 20:27:27
<input type="checkbox"/>	Ubah	Salin	Hapus	2	Adi	Sukorejo	082	2024-01-06 21:50:20	2024-01-06 21:50:20
<input type="checkbox"/>	Ubah	Salin	Hapus	3	Suryo	Majapahit	0882	2024-01-06 21:50:39	2024-01-06 21:50:39
<input type="checkbox"/>	Ubah	Salin	Hapus	4	Eunike	Mojokerto	0912	2024-01-06 21:51:01	2024-01-06 21:51:01
<input type="checkbox"/>	Ubah	Salin	Hapus	5	Handy	Surabaya	083	2024-01-06 21:51:25	2024-01-06 21:51:25

Gambar 3

Di dalam table anggota terdapat id,nama,alamat,no\_telepon. id berserta created dan updated merupakan data bawaan. Data table ini kemudian ditampilkan dalam view Anggota index

Anggota					
<a href="#">Back to Main Menu</a>		<a href="#">Create New Anggota</a>			
ID	Nama	Alamat	No. Telepon	Edit	Delete
1	Nico	Mojokerto	091	<a href="#">Edit</a>	<a href="#">Delete</a>
2	Adi	Sukorejo	082	<a href="#">Edit</a>	<a href="#">Delete</a>
3	Suryo	Majapahit	0882	<a href="#">Edit</a>	<a href="#">Delete</a>
4	Eunike	Mojokerto	0912	<a href="#">Edit</a>	<a href="#">Delete</a>
5	Handy	Surabaya	083	<a href="#">Edit</a>	<a href="#">Delete</a>

Gambar 4

Pada view ini user juga dapat menambahkan anggota dengan menekan button Create New Anggota dan juga user dapat mengedit serta menghapus entry yang sudah ada

Masukan Data Anggota	
Nama	<input type="text"/>
Alamat	<input type="text"/>
No. Telepon	<input type="text"/>
<input type="button" value="Simpan Data"/>	

Gambar 5

Data yang sudah dimasukan dapat langsung menekan tombol simpan data Dimana file akan langsung menuju database

## Edit Anggota

Nama

Alamat

No\_Telepon

Update

Gambar 6

Data yang sudah selesai diedit dapat langsung menekan tombol update. Dimana file akan diupdate dalam database database

### 3. Bedah Code Anggota

Berikut ini merupakan codingan controller untuk anggota

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Anggota;

class AnggotaController extends Controller
{
    public function anggota_index(){
        return view('anggota.index', ['anggota' => $anggota]);
    }

    public function index(){
        $anggota = Anggota::all();
        return view('anggota.index', ['anggota' => $anggota]);
    }

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

    public function store(Request $request){
        $data = $request->validate([
            'nama' => 'required',
```

```

        'alamat'=> 'required',
        'no_telepon'=> 'required',
    ]);
    $new = Anggota::create($data);
    return redirect(route('anggota.index'));
}

public function edit(Anggota $anggota){
    return view('anggota.edit', ['anggota' => $anggota]);
}

public function update(Anggota $anggota, Request $request){
    $data = $request->validate([
        'nama'=> 'required',
        'alamat'=> 'required',
        'no_telepon'=> 'required',
    ]);
    $anggota->update($data);
    return redirect(route('anggota.index'))->with('success', 'Product
Updated succesfully');
}

public function destroy(Anggota $anggota){
    $anggota->delete();
    return redirect(route('anggota.index'))->with('success', 'Product
Deleted succesfully');
}
}

```

Disini terdapat public function Index yang bertujuan menunjukkan data yang sudah ada yang terdapat pada gambar 4. Terdapat function create dan store yang bertujuan dalam pembuatan dan memasukan data baru ke dalam database pada gambar 5. Lalu fungsi edit diggunakan untuk memasukan user ke dalam view edit yang kemudian function update yang digunakan untuk mengupdate data pada gambar 6. Dan terakhir fungsi destroy digunakan dalam system delete untuk menghapus data entry pada gambar 4

Berikut ini merupakan codingan model untuk anggota

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Anggota extends Model
{
    protected $table = 'anggota';
}

```

```

use HasFactory;
protected $fillable = [
    'nama',
    'alamat',
    'no_telepon'
];

public function catalogs()
{
    return $this->belongsToMany(Katalog::class, 'peminjaman',
'anggota_id', 'katalog_id')
        ->withPivot(['tanggal_pinjam', 'tanggal_harus_kembali'])
        ->withTimestamps();
}
}

```

Codingan diatas merupakan bagian Model untuk anggota dimana dapat menentukan protected fillable sesuai dengan kebutuhan yaitu nama, alamat, no\_telepon , lalu dibagian bawah terdapat public function catalogs ini akan berhubungan dengan foreign key untuk peminjaman (id anggota)

Berikut ini merupakan codingan migrations untuk anggota

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
    {
        Schema::create('anggota', function (Blueprint $table) {
            $table->id();
            $table->string('nama');
            $table->string('alamat');
            $table->string('no_telepon');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {

```

```

        Schema::dropIfExists('anggota');
    }
};

```

Pada codingan di atas perlu memasukan table apa yang dibuat dan juga berisikan colom apa saja beserta jenis data apa yang akan dimasukan ke dalamnya

Berikut ini merupakan codingan view index untuk anggota

```

<body>
    <h1>Anggota</h1>
    <div>
        <a href="{{ route('welcome') }}">
            <button>Back to Main Menu</button>
        </a>
        <a href="{{ route('anggota.create') }}">
            <input type="submit" value="Create New Anggota" />
        </a>
    </div>
    <div class="success-message">
        @if(session()->has('success'))
            {{ session('success') }}
        @endif
    </div>
    <div>
        <table>
            <tr>
                <th>ID</th>
                <th>Nama</th>
                <th>Alamat</th>
                <th>No. Telepon</th>
                <th>Edit</th>
                <th>Delete</th>
            </tr>
            @foreach($anggota as $anggota)
                <tr>
                    <td>{{ $anggota->id }}</td>
                    <td>{{ $anggota->nama }}</td>
                    <td>{{ $anggota->alamat }}</td>
                    <td>{{ $anggota->no_telepon }}</td>
                    <td>
                        <a href="{{ route('anggota.edit', ['anggota' =>
$anggota]) }}">Edit</a>
                    </td>
                    <td>
                        <form method="post" action="{{ {
route('anggota.destroy', ['anggota' => $anggota]) }}">
                            @csrf
                            @method('delete')
                            <input type="submit" value="Delete" />

```

```

        </form>
      </td>
    </tr>
  @endforeach
</table>
</div>
</body>

```

Dalam view ini bertujuan untuk menampilkan satu persatu apa yang ada di dalam database anggota lalu menampilkanya. System ini menggunakan foreach untuk memeriksa satu per satu

Berikut ini merupakan codingan view create untuk anggota

```

<body>
  <h1>Masukan Data Anggota</h1>
  <div>
    @if($errors->any())
      <ul>
        @foreach($errors->all() as $error)
          <li>{{$error}}</li>
        @endforeach
      </ul>
    @endif
  </div>
  <form method="post" action="{{ route('anggota.store') }}">
    @csrf
    <div>
      <label>Nama</label>
      <input type="text" name="nama" placeholder="Nama"/>
    </div>
    <div>
      <label>Alamat</label>
      <input type="text" name="alamat" placeholder="Alamat"/>
    </div>
    <div>
      <label>No_Telepon</label>
      <input type="text" name="no_telepon" placeholder="No. Telepon"/>
    </div>
    <div>
      <input type="submit" value="Simpan Data">
    </div>
  </form>
</body>

```

Dalam view ini terdapat banyak inputan data yang perlu diisi sesuai dengan data yang disimpan dalam database anggota dan lalu jika sudah selesai dapat menekan tombol submit untuk menyimpannya kedalam database

Berikut ini merupakan codingan view edit untuk anggota

```

<body>
  <h1>Edit Anggota</h1>

```



```

<div>
    @if($errors->any())
        <ul>
            @foreach($errors->all() as $error)
                <li>{{$error}}</li>
            @endforeach
        </ul>
    @endif
</div>
<form method="post" action="{{ route('anggota.update', ['anggota' =>
$anggota]) }}">
    @csrf
    @method('put')
    <div>
        <label>Nama</label>
        <input type="text" name="nama" placeholder="Nama" value="{{
$anggota->nama }}" />
    </div>
    <div>
        <label>Alamat</label>
        <input type="text" name="alamat" placeholder="Alamat" value="{{
$anggota->alamat }}" />
    </div>
    <div>
        <label>No_Telepon</label>
        <input type="text" name="no_telepon" placeholder="No_Telepon"
value="{{ $anggota->no_telepon }}" />
    </div>
    <div>
        <input type="submit" value="Update" />
    </div>
</form>
</body>

```

Codingan ini tidak jauh beda dengan codingan create. Yang membuat edit unique adalah tambahan codingan value yang akan menandakan value yang baru setelah perubahan. Dan juga hal yang membuatnya beda dengan create terdapat pada controller

Berikut ini adalah route yang dipakai dibelakang yang menghubungkan system anggota

```

//MARK: RouteAnggota
Route::get('/anggota', [AnggotaController::class, 'index'])-
>name('anggota.index');
Route::get('/anggota/create', [AnggotaController::class, 'create'])-
>name('anggota.create');
Route::post('/anggota', [AnggotaController::class, 'store'])-
>name('anggota.store');
Route::get('/anggota/{anggota}/edit', [AnggotaController::class, 'edit'])-
>name('anggota.edit');

```

```
Route::put('/anggota/{anggota}/update', [AnggotaController::class, 'update'])->>name('anggota.update');
Route::delete('/anggota/{anggota}/destroy', [AnggotaController::class, 'destroy'])->>name('anggota.destroy');
```

#### 4. Katalog

←T→			id	judul_buku	pengarang	tahun_terbit	created_at	updated_at	
<input type="checkbox"/>	 Ubah	 Salin	 Hapus	1	Ida sakto	Awang	2001	2024-01-06 20:25:27	2024-01-06 20:25:27
<input type="checkbox"/>	 Ubah	 Salin	 Hapus	2	Ida sakto	Awang	2002	2024-01-06 20:26:36	2024-01-06 20:26:36
<input type="checkbox"/>	 Ubah	 Salin	 Hapus	6	Coding	Yusron	2022	2024-01-06 23:18:37	2024-01-06 23:18:37
<input type="checkbox"/>	 Ubah	 Salin	 Hapus	7	Harry Potter	J K Rowling	1999	2024-01-06 23:18:55	2024-01-06 23:18:55
<input type="checkbox"/>	 Ubah	 Salin	 Hapus	8	Upin Ipin	Joko	2000	2024-01-06 23:19:34	2024-01-06 23:19:34

Gambar 7

Table diatas merupakan table katalog yang Dimana menyimpan judul buku, pengarang dan tahun terbit buku itu

Katalog					
<a href="#">Back to Main Menu</a>		<a href="#">Create New Katalog</a>			
ID	Judul Buku	Pengarang	Tahun Terbit	Edit	Delete
1	Ida sakto	Awang	2001	<a href="#">Edit</a>	<a href="#">Delete</a>
2	Ida sakto	Awang	2002	<a href="#">Edit</a>	<a href="#">Delete</a>
6	Coding	Yusron	2022	<a href="#">Edit</a>	<a href="#">Delete</a>
7	Harry Potter	J K Rowling	1999	<a href="#">Edit</a>	<a href="#">Delete</a>
8	Upin Ipin	Joko	2000	<a href="#">Edit</a>	<a href="#">Delete</a>

Gambar 8

Gambar diatas merupakan bentuk dari index dimana user dapat memasukan data katalog

## Masukan Data Katalog

Judul Buku

Pengarang

Tahun Terbit

Simpan Data

Gambar 9

Data yang sudah dimasukan dapat langsung menekan tombol simpan data Dimana file akan langsung menuju database

## Edit a product

Judul Buku

Pengarang

Tahun Terbit

Update

Gambar 10

Data yang sudah selesai diedit dapat langsung menekan tombol update. Dimana file akan diupdate dalam database database

### 5. Bedah Code Katalog

Berikut ini merupakan codingan controller untuk katalog

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
```

```
use App\Models\Katalog;

class KatalogController extends Controller
{
    public function index()
    {
        $katalog = Katalog::all();
        return view('katalog.index', ['katalog' => $katalog]);
    }

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

    public function store(Request $request)
    {
        $data = $request->validate([
            'judul_buku' => 'required',
            'pengarang' => 'required',
            'tahun_terbit' => 'required',
        ]);
        $new = Katalog::create($data);
        return redirect(route('katalog.index'));
    }

    public function edit(Katalog $katalog){
        return view('katalog.edit', ['katalog' => $katalog]);
    }

    public function update(Katalog $katalog, Request $request){
        $data = $request->validate([
            'judul_buku' => 'required',
            'pengarang' => 'required',
            'tahun_terbit' => 'required',
        ]);
        $katalog->update($data);
        return redirect(route('katalog.index'))->with('success', 'Product
Updated succesfully');
    }
    public function destroy(Katalog $katalog)
    {
        $katalog->delete();
        return redirect(route('katalog.index'))->with('success', 'Product
Deleted successfully');
    }
}
```

Disini terdapat public function Index yang bertujuan menunjukan data yang sudah ada yang terdapat pada gambar 8. Terdapat function create dan store yang bertujuan dalam pembuatan dan memasukan data baru ke dalam database pada gambar 9. Lalu fungsi edit digunakan untuk memasukan user ke dalam view edit yang kemudian function update yang digunakan untuk mengupdate data pada gambar 10. Dan terakhir fungsi destroy digunakan dalam system delete untuk menghapus data entry pada gambar 8

Berikut ini merupakan codingan model untuk katalog

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Katalog extends Model
{
    protected $table = 'katalog';
    use HasFactory;
    protected $fillable = [
        'judul_buku',
        'pengarang',
        'tahun_terbit'
    ];

    public function anggotas()
    {
        return $this->belongsToMany(Anggota::class, 'peminjaman',
        'katalog_id', 'anggota_id')
        ->withPivot(['tanggal_pinjam', 'tanggal_harus_kembali'])
        ->withTimestamps();
    }
}
```

Codingan diatas merupakan bagian Model untuk katalog dimana dapat menentukan protected fillable sesuai dengan kebutuhan yaitu judul\_buku, pengarang, tahun\_terbit, lalu dibagian bawah terdapat public function anggotas ini akan berhubungan dengan foreign key untuk peminjaman (id katalog)

Berikut ini merupakan codingan migrations untuk katalog

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
```

```

    * Run the migrations.
    */
    public function up()
    {
        Schema::create('katalog', function (Blueprint $table) {
            $table->id();
            $table->string('judul_buku');
            $table->string('pengarang');
            $table->string('tahun_terbit');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('katalog');
    }
};

```

Pada codingan di atas perlu memasukan table apa yang dibuat dan juga berisikan colom apa saja beserta jenis data apa yang akan dimasukan ke dalamnya

Berikut ini merupakan codingan view index untuk katalog

```

</head>
<body>
    <h1>Katalog</h1>
    <div>
        <a href="{{ route('welcome') }}">
            <button>Back to Main Menu</button>
        </a>
        <a href="{{ route('katalog.create') }}">
            <input type="submit" value="Create New Katalog" />
        </a>
    </div>
    <div class="success-message">
        @if(session()->has('success'))
            {{ session('success') }}
        @endif
    </div>
    <div>
        <table>
            <tr>
                <th>ID</th>
                <th>Judul Buku</th>
                <th>Pengarang</th>
                <th>Tahun Terbit</th>
            </tr>

```

```

        <th>Edit</th>
        <th>Delete</th>
    </tr>
    @foreach($katalog as $katalog)
        <tr>
            <td>{{ $katalog->id }}</td>
            <td>{{ $katalog->judul_buku }}</td>
            <td>{{ $katalog->pengarang }}</td>
            <td>{{ $katalog->tahun_terbit }}</td>
            <td>
                <a href="{{ route('katalog.edit', ['katalog' =>
$katalog]) }}">Edit</a>
            </td>
            <td>
                <form method="post" action="{{ route('katalog.destroy', ['katalog' => $katalog]) }}">
                    @csrf
                    @method('delete')
                    <input type="submit" value="Delete" />
                </form>
            </td>
        </tr>
    @endforeach
</table>
</div>
</body>
</html>

```

Dalam view ini bertujuan untuk menampilkan satu persatu apa yang ada di dalam database katalog lalu menampilkanya. System ini menggunakan foreach untuk memeriksa satu per satu

Berikut ini merupakan codingan view create untuk katalog

```

<body>
    <h1>Masukan Data Katalog</h1>
    <div>
        @if($errors->any())
            <ul>
                @foreach($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        @endif
    </div>
    <form method="post" action="{{ route('katalog.store') }}">
        @csrf
        <div>
            <label>Judul Buku</label>
            <input type="text" name="judul_buku" placeholder="Judul Buku"/>

```

```

    </div>
    <div>
        <label>Pengarang</label>
        <input type="text" name="pengarang" placeholder="Pengarang"/>
    </div>
    <div>
        <label>Tahun Terbit</label>
        <input type="text" name="tahun_terbit" placeholder="Tahun
Terbit"/>
    </div>
    <div>
        <input type="submit" value="Simpan Data">
    </div>
</form>
</body>

```

Dalam view ini terdapat banyak inputan data yang perlu diisi sesuai dengan data yang disimpan dalam database katalog dan lalu jika sudah selesai dapat menekan tombol submit untuk menyimpannya kedalam database

Berikut ini merupakan codingan view edit untuk katalog

```

<body>
    <h1>Edit a product</h1>
    <div>
        @if($errors->any())
            <ul>
                @foreach($errors->all() as $error)
                    <li>{{$error}}</li>
                @endforeach
            </ul>
        @endif
    </div>
    <form method="post" action="{{ route('katalog.update', ['katalog' =>
$katalog]) }}">
        @csrf
        @method('put')
        <div>
            <label>Judul Buku</label>
            <input type="text" name="judul_buku" placeholder="Judul Buku"
value="{{ $katalog->judul_buku }}" />
        </div>
        <div>
            <label>Pengarang</label>
            <input type="text" name="pengarang" placeholder="Pengarang"
value="{{ $katalog->pengarang }}" />
        </div>
        <div>
            <label>Tahun Terbit</label>

```



```

        <input type="text" name="tahun_terbit" placeholder="Tahun Terbit"
value="{{ $katalog->tahun_terbit }}" />
    </div>
    <div>
        <input type="submit" value="Update" />
    </div>
</form>
</body>

```

Codingan ini tidak jauh beda dengan codingan create. Yang membuat edit unique adalah tambahan codingan value yang akan menandakan value yang baru setelah perubahan. Dan juga hal yang membuatnya beda dengan create terdapat pada controller

Berikut ini adalah route yang dipakai dibelakang yang menghubungkan system katalog

```

Route::get('/katalog', [KatalogController::class, 'index'])-
>name('katalog.index');
Route::get('/katalog/create',[KatalogController:: class,'create'])-
>name('katalog.create');
Route::post('/katalog', [KatalogController::class, 'store'])-
>name('katalog.store');
Route::get('/katalog/{katalog}/edit', [KatalogController::class, 'edit'])-
>name('katalog.edit');
Route::put('/katalog/{katalog}/update', [KatalogController::class, 'update'])-
>name('katalog.update');
Route::delete('/katalog/{katalog}/destroy', [KatalogController::class,
'destroy'])->name('katalog.destroy');

```

## 6. Peminjaman

<div>← T →</div>			▼ id	anggota_id	katalog_id	tanggal_pinjam	tanggal_harus_kembali	created_at	updated_at	
<input type="checkbox"/>	<div><div>Ubah</div><div><div><div></div><div></div><div></div></div></div></div>	<div><div>Salin</div><div><div></div><div></div><div></div></div></div>	<div><div>Hapus</div><div><div></div><div></div><div></div></div></div>	1	1	1	9999-09-09	9999-09-16	2024-01-06 20:27:41	2024-01-06 20:27:41
<input type="checkbox"/>	<div><div>Ubah</div><div><div></div><div></div><div></div></div></div>	<div><div>Salin</div><div><div></div><div></div><div></div></div></div>	<div><div>Hapus</div><div><div></div><div></div><div></div></div></div>	2	3	6	2001-12-12	2001-12-19	2024-01-07 00:21:41	2024-01-07 00:21:41
<input type="checkbox"/>	<div><div>Ubah</div><div><div></div><div></div><div></div></div></div>	<div><div>Salin</div><div><div></div><div></div><div></div></div></div>	<div><div>Hapus</div><div><div></div><div></div><div></div></div></div>	3	4	7	2013-02-03	2013-02-10	2024-01-07 00:34:03	2024-01-07 00:34:03
<input type="checkbox"/>	<div><div>Ubah</div><div><div></div><div></div><div></div></div></div>	<div><div>Salin</div><div><div></div><div></div><div></div></div></div>	<div><div>Hapus</div><div><div></div><div></div><div></div></div></div>	4	4	8	2005-12-22	2005-12-29	2024-01-07 00:34:27	2024-01-07 00:34:27
<input type="checkbox"/>	<div><div>Ubah</div><div><div></div><div></div><div></div></div></div>	<div><div>Salin</div><div><div></div><div></div><div></div></div></div>	<div><div>Hapus</div><div><div></div><div></div><div></div></div></div>	5	3	2	2006-06-06	2006-06-13	2024-01-07 00:35:09	2024-01-07 00:35:09

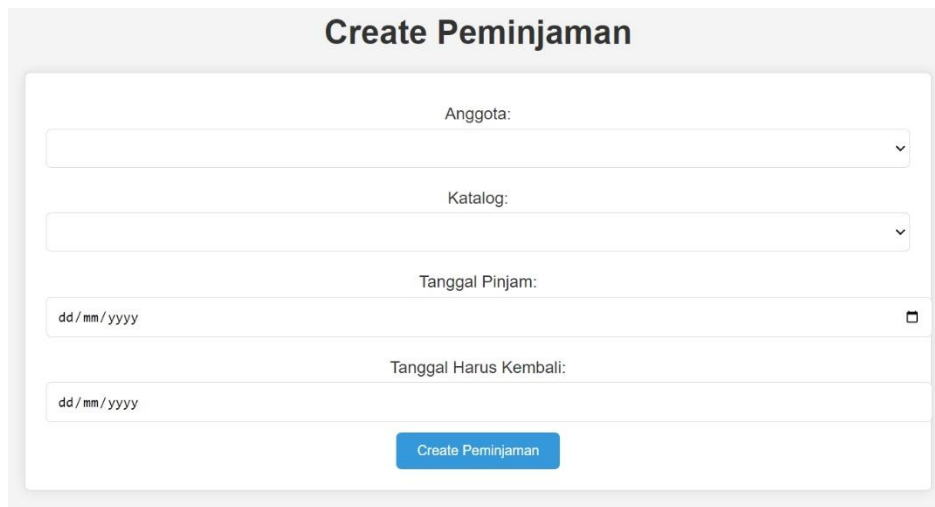
Gambar 11

Table diatas merupakan table katalog yang Dimana menyimpan 2 foreign key yaitu id dari anggota dan id dari katalog beserta tanggal peminjaman dan tanggal harus Kembali

Peminjaman List					
<div>Back to Main Menu</div> <div>Create Peminjaman</div>					
ID	Anggota ID	Katalog ID	Tanggal Pinjam	Tanggal Harus Kembali	Action
1	1	1	9999-09-09	9999-09-16	<div>Edit</div> <div>Delete</div>
2	3	6	2001-12-12	2001-12-19	<div>Edit</div> <div>Delete</div>
3	4	7	2013-02-03	2013-02-10	<div>Edit</div> <div>Delete</div>
4	4	8	2005-12-22	2005-12-29	<div>Edit</div> <div>Delete</div>
5	3	2	2006-06-06	2006-06-13	<div>Edit</div> <div>Delete</div>

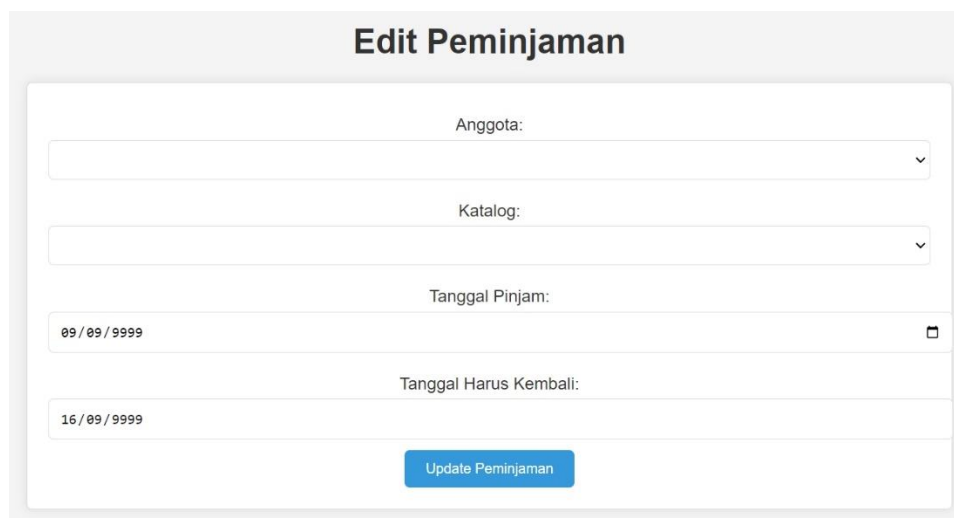
Gambar 12

Gambar diatas merupakan bentuk dari index dimana user dapat memasukan data peminjaman



Gambar 13

Data yang sudah dimasukan dapat langsung menekan tombol simpan data Dimana file akan langsung menuju database



Gambar 14

Data yang sudah selesai diedit dapat langsung menekan tombol update. Dimana file akan diupdate dalam database database

## 7. Bedah Code Peminjaman

Berikut ini merupakan codingan controller untuk peminjaman

```
<?php
namespace App\Http\Controllers;
use Illuminate\Validation\Rule;
use Illuminate\Http\Request;
```

```

use App\Models\Peminjaman;
use App\Models\Anggota;
use App\Models\Katalog;

class PeminjamanController extends Controller
{
    public function index()
    {
        $peminjamans = Peminjaman::all();
        return view('peminjaman.index', compact('peminjamans'));
    }

    public function create()
    {
        $anggotas = Anggota::all();
        $katalogs = Katalog::all();

        return view('peminjaman.create', compact('anggotas', 'katalogs'));
    }

    public function store(Request $request)
    {
        // Validate the request
        $request->validate([
            'anggota_id' => 'required',
            'katalog_id' => [
                'required',
                Rule::unique('peminjaman')->where(function ($query) use
($request) {
                    return $query->where('katalog_id', $request->katalog_id);
                }),
            ],
            'tanggal_pinjam' => 'required|date',
            'tanggal_harus_kembali' => 'required|date',
        ]);

        // Create a new Peminjaman record
        Peminjaman::create([
            'anggota_id' => $request->anggota_id,
            'katalog_id' => $request->katalog_id,
            'tanggal_pinjam' => $request->tanggal_pinjam,
            'tanggal_harus_kembali' => $request->tanggal_harus_kembali,
        ]);

        return redirect()->route('peminjaman.index')->with('success',
'Peminjaman created successfully!');
    }
}

```

```

    public function edit($id)
    {
        $peminjaman = Peminjaman::find($id);

        if (!$peminjaman) {
            abort(404, 'Peminjaman not found');
        }
        $anggota = Anggota::all();
        $katalogs = Katalog::all();

        return view('peminjaman.edit', [
            'peminjaman' => $peminjaman,
            'anggota' => $anggota,
            'katalogs' => $katalogs, // Corrected variable name
        ]);
    }
    public function destroy($id)
    {
        // Find the Peminjaman model instance by ID and delete it
        Peminjaman::destroy($id);

        // Redirect back or to any other page as needed
        return redirect()->route('peminjaman.index')->with('success', 'Peminjaman
deleted successfully');
    }
}

```

Disini terdapat public function Index yang bertujuan menunjukan data yang sudah ada yang terdapat pada gambar 12. Terdapat function create dan store yang bertujuan dalam pembuatan dan memasukan data baru ke dalam database pada gambar 13. Lalu fungsi edit diggunakan untuk memasukan user ke dalam view edit yang kemudian function update yang digunakan untuk mengupdate data pada gambar 14. Dan terakhir fungsi destroy digunakan dalam system delete untuk menghapus data entry pada gambar 12. Disini juga ada tambahan fitur jika ada member yang melakukan peminjaman kepada id katalog yang sama maka tidak akan bisa diinput

Berikut ini merupakan codingan model untuk peminjaman

```

<?php

// app/Models/Peminjaman.php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Peminjaman extends Model
{
    protected $table = 'peminjaman'; // Specify the table name if different
from the default
}

```

```

protected $fillable = [
    'anggota_id',
    'katalog_id',
    'tanggal_pinjam',
    'tanggal_harus_kembali',
];

public function anggota()
{
    return $this->belongsTo(Anggota::class, 'anggota_id');
}

public function katalog()
{
    return $this->belongsTo(Katalog::class, 'katalog_id');
}
}

```

Codingan diatas merupakan bagian Model untuk peminjaman dimana dapat menentukan protected fillable sesuai dengan kebutuhan yaitu anggota id, katalog id, tanggal pinjam, tanggal harus kembali, lalu dibagian bawah terdapat public function anggotas dan kayalogs ini akan berhubungan dengan foreign key untuk peminjaman mengambil id dari katalog dan anggota

Berikut ini merupakan codingan migrations untuk katalog

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('peminjaman', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('anggota_id');
            $table->foreign('anggota_id')->references('id')->on('anggota');
            $table->unsignedBigInteger('katalog_id');
            $table->foreign('katalog_id')->references('id')->on('katalog');
            $table->string('tanggal_pinjam');
            $table->string('tanggal_harus_kembali');
            $table->timestamps();
        });
    }
}

```

```

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::dropIfExists('peminjaman');
}
};

```

Pada codingan di atas perlu memasukan table apa yang dibuat dan juga berisikan colom apa saja beserta jenis data apa yang akan dimasukan ke dalamnya

Berikut ini merupakan codingan view index untuk katalog

```

<body>
    <h1>Peminjaman List</h1>
    <a class="main-menu-button" href="{{ route('welcome') }}">Back to Main
Menu</a>
    <div>
        <a class="button" href="{{ route('peminjaman.create') }}">Create
Peminjaman</a>
    </div>

    <table>
        <tr>
            <th>ID</th>
            <th>Anggota ID</th>
            <th>Katalog ID</th>
            <th>Tanggal Pinjam</th>
            <th>Tanggal Harus Kembali</th>
            <th>Action</th>
        </tr>
        @foreach ($peminjamans as $peminjaman)
            <tr>
                <td>{{ $peminjaman->id }}</td>
                <td>{{ $peminjaman->anggota_id }}</td>
                <td>{{ $peminjaman->katalog_id }}</td>
                <td>{{ $peminjaman->tanggal_pinjam }}</td>
                <td>{{ $peminjaman->tanggal_harus_kembali }}</td>
                <td>
                    <a class="button" href="{{ route('peminjaman.edit',
$peminjaman->id) }}">Edit</a>
                    <form method="post" action="{{ route('peminjaman.destroy',
$peminjaman->id) }}" style="display: inline-block;">
                        @csrf
                        @method('DELETE')

```

```

                                <button type="submit" class="delete-button"
onclick="return confirm('Are you sure you want to delete this
item?')">Delete</button>
                                </form>
                                </td>
                                </tr>
                                @endforeach
                                </table>
</body>

```

Dalam view ini terdapat banyak inputan data yang perlu diisi sesuai dengan data yang disimpan dalam database penyimpanan dan lalu jika sudah selesai dapat menekan tombol submit untuk menyimpannya kedalam database

Berikut ini merupakan codingan view create untuk katalog

```

<body>
    <h1>Create Peminjaman</h1>

    <form method="post" action="{{ route('peminjaman.store') }}">
        @csrf

        <label for="anggota_id">Anggota:</label>
        <select name="anggota_id" id="anggota_id">
            @foreach ($anggotas as $anggota)
                <option value="{{ $anggota->id }}">{{ $anggota->name
}}</option>
            @endforeach
        </select>

        <label for="katalog_id">Katalog:</label>
        <select name="katalog_id" id="katalog_id">
            @foreach ($katalogs as $katalog)
                <option value="{{ $katalog->id }}">{{ $katalog->title
}}</option>
            @endforeach
        </select>

        <label for="tanggal_pinjam">Tanggal Pinjam:</label>
        <input type="date" name="tanggal_pinjam" id="tanggal_pinjam">

        <label for="tanggal_harus_kembali">Tanggal Harus Kembali:</label>
        <input type="date" name="tanggal_harus_kembali"
id="tanggal_harus_kembali" readonly>

        <button type="submit">Create Peminjaman</button>
    </form>
</body>

```

Dalam view ini terdapat banyak inputan data yang perlu diisi sesuai dengan data yang disimpan dalam database katalog dan lalu jika sudah selesai dapat menekan tombol submit untuk menyimpannya kedalam database

Berikut ini merupakan codingan view edit untuk katalog

```
<body>
  <h1>Edit Peminjaman</h1>

  <form method="post" action="{ route('peminjaman.update', $peminjaman->id)
  }}">
    @csrf
    @method('PUT')

    <label for="anggota_id">Anggota:</label>
    <select name="anggota_id" id="anggota_id">
      @foreach ($anggota as $anggotaItem)
        <option value="{ $anggotaItem->id }" @if($anggotaItem->id ==
        $peminjaman->anggota_id) selected @endif>{{ $anggotaItem->name }}</option>
      @endforeach
    </select>

    <label for="katalog_id">Katalog:</label>
    <select name="katalog_id" id="katalog_id">
      @foreach ($katalogs as $katalog)
        <option value="{ $katalog->id }" @if($katalog->id ==
        $peminjaman->katalog_id) selected @endif>{{ $katalog->title }}</option>
      @endforeach
    </select>

    <label for="tanggal_pinjam">Tanggal Pinjam:</label>
    <input type="date" name="tanggal_pinjam" id="tanggal_pinjam" value="{ {
    $peminjaman->tanggal_pinjam } }">

    <label for="tanggal_harus_kembali">Tanggal Harus Kembali:</label>
    <input type="date" name="tanggal_harus_kembali"
    id="tanggal_harus_kembali" value="{ { $peminjaman->tanggal_harus_kembali } }"
    readonly>

    <button type="submit">Update Peminjaman</button>
  </form>
</body>
```

Codingan ini tidak jauh beda dengan codingan create. Yang membuat edit unique adalah tambahan codingan value yang akan menandakan value yang baru setelah perubahan. Dan juga hal yang membuatnya beda dengan create terdapat pada controller

Berikut ini adalah route yang dipakai dibelakang yang menghubungkan system peminjaman

```
Route::get('/peminjaman', [PeminjamanController::class, 'index'])-
>name('peminjaman.index');
```



```
Route::get('/peminjaman/create', [PeminjamanController::class, 'create'])-  
>name('peminjaman.create');  
Route::post('/peminjaman', [PeminjamanController::class, 'store'])-  
>name('peminjaman.store');  
Route::get('/peminjaman/{id}', [PeminjamanController::class, 'show'])-  
>name('peminjaman.show');  
Route::get('/peminjaman/{id}/edit', [PeminjamanController::class, 'edit'])-  
>name('peminjaman.edit');  
Route::put('/peminjaman/{id}', [PeminjamanController::class, 'update'])-  
>name('peminjaman.update');  
Route::delete('/peminjaman/{id}', [PeminjamanController::class, 'destroy'])-  
>name('peminjaman.destroy');
```