

LAPORAN PRAKTIKUM PEMROGRAMAN WEB
RESPONSI



Disusun oleh:

Nama : Afif Imam Rahadi

Nim : L0122006

Kelas : A

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

2024

1. Screenshot Source Code

Model Student

```
<?php

namespace App\Models;

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

class Student extends Model
{
    use HasFactory;
    protected $fillable = [
        'nama', 'nim', 'prodi', 'kelas', 'angkatan'
    ];
}
```

Kode di atas adalah code bagian model Student. Model ini mewakili tabel "students" dalam database dan digunakan untuk berinteraksi dengan data di tabel tersebut. Terdapat beberapa import yang diperlukan agar code ini dapat berjalan. Class Student extends Model mendefinisikan model Student yang mewarisi fitur-fitur dari kelas Model. Trait HasFactory digunakan untuk membuat instance model dalam pengujian atau seeding database. Properti protected \$fillable adalah array yang berisi atribut-atribut yang dapat diisi, yaitu nama, nim, prodi, kelas, dan angkatan. Atribut-atribut ini menunjukkan kolom-kolom dalam tabel yang dapat diisi oleh pengguna melalui metode mass assignment.

Model User

```
namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'username',
        'email',
        'password',
    ];
}
```

Kode di atas adalah code dari model User. Model ini berada di namespace App\Models dan mewarisi kelas Authenticatable, yang merupakan bagian dari sistem autentikasi Laravel. Properti dari model ini adalah protected \$fillable menentukan atribut yang dapat diisi, seperti name, username, email, dan password. Properti protected \$hidden menentukan atribut yang

harus disembunyikan saat model diserialisas, yaitu password dan remember_token. Metode casts mengonversi atribut tertentu ke tipe data yang sesuai saat diakses, seperti mengonversi email_verified_at menjadi objek datetime dan password menjadi hashed. Kode ini memastikan keamanan dan integritas data pengguna dengan mengatur atribut yang dapat diisi, disembunyikan, dan dikonversi secara otomatis.

Route

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\LoginController;
use App\Http\Controllers\StudentController;
use App\Http\Controllers\RegisterController;

Route::get('/', function () {
    return view('welcome');
}) -> name('welcome');

Route::get('/students', [StudentController::class, 'index'])->middleware('auth')-> name('students');

Route::get('/students/form', [StudentController::class, 'create']);

Route::post('/students/add', [StudentController::class, 'store']) -> name('add-students');
Route::get('/students/delete/{id}', [StudentController::class, 'destroy']);
Route::get('/students/edit/{student}', [StudentController::class, 'edit']);
Route::post('/students/update/{student}', [StudentController::class, 'update']) -> name('students.update');

// Login
Route::get('/login', [LoginController::class, 'index'])->name('login');
Route::post('/login', [LoginController::class, 'authenticate']);

// Logout
Route::post('/logout', [LoginController::class, 'logout']);

// register
Route::get('/register', [RegisterController::class, 'index']);
Route::post('/register', [RegisterController::class, 'store']);
```

Kode di atas adalah code untuk routing pada web ini. Router ini mengatur berbagai jalur HTTP dan mengaitkannya dengan controller serta metode yang sesuai. Route get('/') mengarahkan ke halaman utama yang memuat tampilan 'welcome'. Route get('/students') mengarahkan ke metode index di StudentController dan hanya dapat diakses jika pengguna terautentikasi (dengan middleware 'auth'). Route get('/students/form') memuat formulir untuk menambahkan mahasiswa, dan route post('/students/add') mengarahkan ke metode store untuk menyimpan data mahasiswa baru. Route get('/students/delete/{id}') dan get('/students/edit/{student}')

masing-masing mengarahkan ke metode destroy dan edit untuk menghapus atau mengedit data mahasiswa berdasarkan ID atau model student yang diteruskan. Route post('/students/update/{student}')

mengarahkan ke metode update untuk memperbarui data mahasiswa. Untuk autentikasi, route get('/login') dan post('/login') mengarahkan ke metode index dan authenticate di LoginController untuk memuat halaman login dan memproses login. Route post('/logout') mengarahkan ke metode logout untuk keluar dari sesi. Untuk pendaftaran, route get('/register') dan post('/register') mengarahkan ke metode index dan store di RegisterController untuk memuat halaman registrasi dan menyimpan data pengguna baru.

Student Controller

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Student;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\DB;
8
9 class StudentController extends Controller
10 {
11     /**
12      * Display a listing of the resource.
13      */
14     public function index()
15     {
16         $students = DB::table('students')->get();
17         return view('student.index', ['students' => $students]);
18     }
19
20     /**
21      * Show the form for creating a new resource.
22      */
23     public function create()
24     {
25         //
26     }
27
28     /**
29      * Store a newly created resource in storage.
30      */
31     public function store(Request $request)
32     {
33         $request->validate([
34             'nama' => 'required',
35             'nim' => 'required',
36             'prodi' => 'required',
37             'kelas' => 'required',
38             'angkatan' => 'required',
39         ]);
40
41         $student = new Student;
42         // kiri objek model - kanan dari request
43         $student->nama = $request->nama;
44         $student->nim = $request->nim;
45         $student->prodi = $request->prodi;
46         $student->kelas = $request->kelas;
47         $student->angkatan = $request->angkatan;
48
49         $student->save();
50
51         return redirect()->route('students')->with('success', 'Data mahasiswa berhasil ditambahkan');
52     }
53
54     /**
55      * Display the specified resource.
56      */
57     public function show(Student $student)
58     {
59         //
60     }
61
62     /**
63      * Show the form for editing the specified resource.
64      */
65     public function edit(Student $student)
66     {
67         return view('student.edit', ['student' => $student]);
68     }
69
70     /**
71      * Update the specified resource in storage.
72      */
73     public function update(Request $request, Student $student)
74     {
75         $request->validate([
76             'nama' => 'required',
77             'nim' => 'required',
78             'prodi' => 'required',
79             'kelas' => 'required',
80             'angkatan' => 'required',
81         ]);
82
83         $student->update($request->all());
84         return redirect()->route('students')->with('success', 'Data berhasil diupdate');
85     }
86
87     /**
88      * Remove the specified resource from storage.
89      */
90     public function destroy($id)
91     {
92         $student = DB::table('students')->where('id', $id)->delete();
93         return redirect()->route('students');
94     }
95 }
```

Code ini adalah definisi dari StudentController di Laravel yang mengatur logika aplikasi terkait operasi CRUD (Create, Read, Update, Delete) untuk model Student. Controller ini berada di dalam namespace App\Http\Controllers yang menunjukkan bahwa file ini berada di direktori app/Http/Controllers. Beberapa kelas diimpor di bagian awal, termasuk Student dari App\Models, Request dari Illuminate\Http\Request, dan DB dari Illuminate\Support\Facades.

Metode index dalam controller ini mengambil semua data dari tabel students menggunakan query builder `DB::table('students')->get()`, dan mengirimkannya ke view `student.index`. Metode create didefinisikan tetapi kosong, menunjukkan bahwa form untuk membuat resource baru belum diimplementasikan. Metode store menerima Request dari form, memvalidasi input untuk memastikan semua kolom wajib diisi (nama, nim, prodi, kelas, dan angkatan), kemudian membuat instance baru dari Student dan menyimpan data yang valid ke dalam database. Setelah data berhasil disimpan, pengguna akan dialihkan ke route students dengan pesan sukses.

Metode show didefinisikan tetapi kosong, biasanya digunakan untuk menampilkan detail dari satu resource Student. Metode edit menerima instance Student sebagai parameter, mengembalikan view `student.edit` dengan data mahasiswa yang akan diedit. Metode update menerima Request dan instance Student, memvalidasi input, kemudian memperbarui data mahasiswa yang ada dengan data baru dari request. Setelah berhasil memperbarui data, pengguna akan dialihkan kembali ke route students dengan pesan sukses.

Metode destroy menerima parameter `$id`, menghapus data mahasiswa dari tabel students berdasarkan id yang diberikan, kemudian mengarahkan pengguna kembali ke route students. Metode ini menggunakan query builder untuk operasi penghapusan. Secara keseluruhan, controller ini menangani semua operasi dasar CRUD untuk model Student, memastikan data dapat dibuat, dibaca, diperbarui, dan dihapus sesuai kebutuhan aplikasi.

Migration

```
<?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('students', function (Blueprint $table) {
            $table->id();
            $table->string('nama');
            $table->string('nim');
            $table->string('prodi');
            $table->char('kelas');
            $table->integer('angkatan');
            $table->timestamps();
        });
    }
}
```

Kode ini adalah definisi migrasi database di Laravel yang digunakan untuk membuat dan menghapus tabel students. Migrasi ini terletak dalam namespace `Illuminate\Database\Migrations` dan menggunakan beberapa kelas dari Laravel termasuk `Blueprint` dan `Schema`. Kode ini didefinisikan dalam sebuah anonymous class yang mewarisi `Migration`.

Metode up didefinisikan untuk menjalankan migrasi, di mana Schema::create digunakan untuk membuat tabel students. Di dalam fungsi ini, sebuah instance Blueprint diberikan sebagai parameter untuk mendefinisikan struktur tabel. Tabel students memiliki beberapa kolom: id yang merupakan primary key dengan auto-increment, nama dan nim yang keduanya adalah string, prodi yang juga merupakan string, kelas yang merupakan karakter (char), angkatan yang merupakan integer, dan dua kolom timestamp (created_at dan updated_at) yang secara otomatis dikelola oleh Laravel. Metode down didefinisikan untuk membalikkan migrasi, yang dalam hal ini menghapus tabel students jika ada. Metode ini menggunakan Schema::dropIfExists untuk memastikan bahwa tabel students dihapus dengan aman.

Welcome Views

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>welcome</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QwTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRj"
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7"
  <style>
    body
    {
      background-color: #007bff;
    }
    .centered {
      height: 100vh;
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="container centered">
    <h1>Selamat Datang Dalam Web Pendataan Mahasiswa</h1>
    <div class="mt-4">
      <button class="btn btn-primary mx-2">
        <a href="/students" class="text-light" style="text-decoration: none;">List Daftar Mahasiswa</a>
      </button>
      <button class="btn btn-primary mx-2">
        <a href="/students/form" class="text-light" style="text-decoration: none;">Tambah Data Mahasiswa</a>
      </button>
    </div>
  </div>
</body>
</html>
```

Pada bagian welcome ini, merupakan teks html biasa yang menggunakan bootstrap sebagai bantuan styling. Pada halaman ini hanya sebagai halaman utama sebelum user dapat melihat dan mengisi data mahasiswa. Di halaman ini, terdapat 2 button yang mengarah ke halaman yang berbeda.

Student/index view

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tampil Data</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXhJ" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9FBDZLSEaAA55N5l5P11XXh21BB" crossorigin="anonymous"></script>
  <style type="text/css">
    .container
    {
      width: 100%;
      margin: auto;
    }
    body
    {
      margin: 8px;
      padding: 10px;
    }
    .tambah-data{
      width: 130px;
      margin-left: 187px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2 class="my-4" style="text-align:center;">Data Mahasiswa</h2>
    <table class="table table-striped table-bordered"
      <tr bgcolor="beige" align="center">
        <th>No</th>
        <th>Nama</th>
        <th>NIM</th>
        <th>Podi</th>
        <th>Kelass</th>
        <th>Angkatan</th>
        <th>Aksi</th>
      </tr>
      <tbody>
        <tr>
          <td>1</td>
          <td>John Doe</td>
          <td>123456789</td>
          <td>10</td>
          <td>1</td>
          <td>2023</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>2</td>
          <td>Jane Smith</td>
          <td>987654321</td>
          <td>20</td>
          <td>2</td>
          <td>2024</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>3</td>
          <td>Bob Johnson</td>
          <td>567890123</td>
          <td>30</td>
          <td>3</td>
          <td>2025</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>4</td>
          <td>Alice Brown</td>
          <td>456789012</td>
          <td>40</td>
          <td>4</td>
          <td>2026</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>5</td>
          <td>Charlie Davis</td>
          <td>345678901</td>
          <td>50</td>
          <td>5</td>
          <td>2027</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>6</td>
          <td>David Wilson</td>
          <td>234567890</td>
          <td>60</td>
          <td>6</td>
          <td>2028</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>7</td>
          <td>Eve Miller</td>
          <td>123456789</td>
          <td>70</td>
          <td>7</td>
          <td>2029</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>8</td>
          <td>Frank Moore</td>
          <td>012345678</td>
          <td>80</td>
          <td>8</td>
          <td>2030</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>9</td>
          <td>Grace Taylor</td>
          <td>901234567</td>
          <td>90</td>
          <td>9</td>
          <td>2031</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
        <tr>
          <td>10</td>
          <td>Henry White</td>
          <td>890123456</td>
          <td>00</td>
          <td>0</td>
          <td>2032</td>
          <td><button class="btn btn-primary">Edit</button><button class="btn btn-danger">Delete</button></td>
        </tr>
      </tbody>
    </table>
    <div class="d-flex justify-content-end align-items-center gap-2">
      <button class="btn btn-primary">Tambah</button>
      <button class="btn btn-danger">Hapus</button>
    </div>
  </div>
</body>
</html>
```

Ini adalah halaman utama dari students. Halaman utama ini akan menampilkan data mahasiswa yang berbentuk tabel. Di dalam tabel ini terdapat data-data yang telah diinputkan oleh user. Pada setiap data terdapat aksi yang dapat dilakukan oleh user, yaitu edit dan delete. Oleh karena itu, user dapat mengatur data sesuai yang diinginkan.

Student/form View

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pHlyT2bRjXKo" crossorigin="anonymous">
  <title style="text-align: center;">Form Data Mahasiswa</title>
</head>
<body>
  <div class="my-4 mx-3">Form Data Mahasiswa</div>
  <div class="container w-75 m-4 mx-5">
    <form method="POST" action="{ route('add-students') }">
      <@csrf>
      <div>
        <div>
          <label class="input-group-text w-25">Nama</label>
          <input class="input-group-text w-25" type="text" name="nama" size="30" placeholder="Masukkan nama anda" >
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">NIM</label>
          <input class="input-group-text w-25" type="text" name="nim" size="30" placeholder="Masukkan nim anda"></input>
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Prodi</label>
          <input class="input-group-text w-25" type="text" name="prodi" size="30" placeholder="Masukkan prodi anda">
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Kelas</label>
          <input class="input-group-text w-25" type="text" name="kelas" size="30" placeholder="Masukkan kelas anda">
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

Ini adalah code halaman dari form data mahasiswa. Di sini user dapat mengisi data sesuai dengan tempatnya. Terdapat tag form agar lebih terstruktur untuk user input. Setiap form sudah memiliki name masing-masing yang nantinya akan diproses untuk dimasukkan ke dalam database.

Student/edit view

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh" crossorigin="anonymous">
  <title style="text-align: center;">Edit Form Data Mahasiswa</title>
</head>
<body>
  <h1 class="my-4 mx-3">Edit Form Data Mahasiswa</h1>
  <div id="mainContainer" class="container w-75 m-4 mx-5">
    <form method="POST" action={{ route('students.update', $student -> id )}}>
      @csrf
      <div>
        <div>
          <label class="input-group-text w-25">Nama</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nama" value="{{ $student->nama }}" size="30" >
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">NIM</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="nim" value="{{ $student->nim }}" size="30"></input>
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Prodi</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="prodi" value="{{ $student->prodi }}" size="30">
        </div>
      </div>
      <div>
        <div>
          <label class="input-group-text w-25 mt-3">Kelas</label>
        </div>
        <div>
          <input class="input-group-text w-25" type="text" name="kelas" value="{{ $student->kelas }}" size="30">
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

Tidak berbeda jauh dengan views form. Yang membedakannya adalah pada halaman edit ini, pada bagian form sudah memiliki value masing-masing yang diambil dari databasenya. Sehingga saat user ingin melakukan edit, sudah ada data sebelumnya sehingga tidak bingung untuk mengganti yang bagian mana.

Login Controller

```
class LoginController extends Controller
{
    public function index()
    {
        return view('login.index');
    }

    public function authenticate(Request $request)
    {
        $credentials = $request->validate([
            'username' => 'required',
            'password' => 'required'
        ]);

        if(Auth::attempt($credentials))
        {
            $request->session()->regenerate();
            return redirect()->intended('/students');
        }

        return back()->with('loginError', 'Login Failed');
    }

    public function logout(Request $request)
    {
        Auth::logout();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

Pada bagian awal terdapat namespace dari controller ini, yaitu App\Http\Controllers. Kemudian, ada dua import statement yang mengimpor Request dari Illuminate\Http\Request dan Auth dari Illuminate\Support\Facades\Auth. Selanjutnya, Kelas LoginController mewarisi dari Controller dasar Laravel. Kelas ini memiliki tiga metode utama: index, authenticate, dan logout. Metode index(), Metode ini hanya mengembalikan view login.index.

Berikutnya terdapat Metode authenticate(Request \$request) yang menerima objek Request sebagai parameter, yang mewakili permintaan HTTP. Pertama, data dari permintaan divalidasi untuk memastikan bahwa username dan password ada dan diperlukan. Setelah validasi, kredensial ini digunakan untuk mencoba autentikasi pengguna menggunakan Auth::attempt(\$credentials). Jika autentikasi berhasil, sesi pengguna diregenerasi untuk mencegah serangan sesi (session fixation attack) dan pengguna diarahkan ke halaman /students. Jika autentikasi gagal, pengguna akan dikembalikan ke halaman login dengan pesan error 'Login Failed'.

Terakhir, terdapat Metode logout(Request \$request) yang menerima objek Request sebagai parameter. Metode ini mengeluarkan pengguna yang sedang login menggunakan Auth::logout(), menginvalidkan sesi untuk menghapus semua data sesi, dan meregenerasi token sesi untuk mencegah serangan CSRF (Cross-Site Request Forgery). Terakhir, pengguna diarahkan ke halaman utama.

Login Index

```
<body>
  <div class="container">
    @if (session()-has('success'))
      <div class="alert alert-success alert-dismissible fade show" role="alert">
        {{ session('success') }}
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
      </div>
    @endif

    @if (session()-has('loginError'))
      <div class="alert alert-danger alert-dismissible fade show" role="alert">
        {{ session('loginError') }}
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
      </div>
    @endif

    <h2 class="my-4" style="text-align:center;">Login</h2>
    <div class="row justify-content-center">
      <div class="col-lg-5">
        <main class="form-signin w-100 m-auto">
          <form action="/login" method="POST">
            @csrf
            <div class="form-floating">
              <input type="text" name="username" class="form-control @error('username') is-invalid @enderror" id="name" placeholder="username" autofocus required value="{{ old('username') }}">
              <label for="name">Username</label>
              @error('username')
                <div class="invalid-feedback">
                  {{ $message }}
                </div>
              @enderror
            </div>
            <div class="form-floating">
              <input type="password" name="password" class="form-control" id="password" placeholder="Password" required>
              <label for="password">Password</label>
            </div>
            <button class="btn btn-primary w-100 py-2" type="submit">Login</button>
          </form>
          <small class="d-block text-center mt-3">Didn't Have An Account? <a href="/register">Register</a></small>
        </main>
      </div>
    </div>
  </div>
```

Ini adalah code html yang digunakan untuk membuat halaman login. Pada Bagian `<body>` berisi elemen `<div class="container">` untuk membungkus konten halaman. Selanjutnya terdapat pengecekan kondisi menggunakan `@if` untuk menampilkan pesan sukses atau kesalahan login berdasarkan kondisi session. Jika ada pesan sukses, sebuah alert Bootstrap dengan kelas `alert-success` ditampilkan; jika ada pesan kesalahan login, sebuah alert dengan kelas `alert-danger` ditampilkan.

Selanjutnya, sebuah judul dan elemen div yang digunakan untuk container dan column untuk form login. Form login diimplementasikan menggunakan atribut `action="/login"` untuk mengirimkan data ke URL `/login` dengan metode POST. Token CSRF disertakan menggunakan `@csrf` untuk keamanan. Form terdiri dari dua input, yaitu `username` dan `password`. Setiap input memiliki validasi Laravel untuk menampilkan pesan kesalahan jika ada, ditandai dengan kelas `is-invalid` dan elemen div dengan kelas `invalid-feedback` yang menampilkan pesan kesalahan. Terakhir, sebuah tombol submit yang digunakan untuk mengirimkan formulir, dan sebuah link untuk pendaftaran pengguna baru ditampilkan di bawah tombol submit.

Register Controller

```
<?php

namespace App\Http\Controllers;

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

class RegisterController extends Controller
{
    public function index()
    {
        return view('register.index');
    }

    public function store(Request $request)
    {
        $validated_data = $request->validate([
            'name' => 'required|max:255',
            'username' => 'required|min:3|max:255|unique:users',
            'email' => 'required|email:dns|unique:users',
            'password' => 'required'
        ]);

        $validated_data['password'] = bcrypt($validated_data['password']);

        User::create($validated_data);
        return redirect('/login')->with('success', 'Registration success, Please Login!');
    }
}
```

Bagian awal terdapat Namespace dan Imports yang digunakan untuk controller. Selanjutnya, terdapat Kelas RegisterController yang memiliki dua metode Utama yaitu, index dan store. Metode index() mengembalikan view register.index. Metode store(Request \$request) menerima objek Request sebagai parameter, yang mewakili permintaan HTTP. Data yang diterima dari permintaan divalidasi menggunakan metode validate(), yang memastikan bahwa nama diperlukan dan memiliki panjang maksimum 255 karakter, username diperlukan dengan panjang minimal 3 dan maksimal 255 karakter serta unik di tabel users, email diperlukan, harus sesuai format DNS, dan juga unik di tabel users, serta password diperlukan. Setelah validasi, password yang diterima di-hash menggunakan bcrypt untuk keamanan. Data yang telah divalidasi dan di-hash kemudian disimpan ke database dengan menggunakan metode create() dari model User. Setelah pengguna berhasil dibuat, pengguna diarahkan ke halaman login (/login) dengan pesan sukses 'Registration success, Please Login!' menggunakan with() untuk menambahkan pesan ke session.

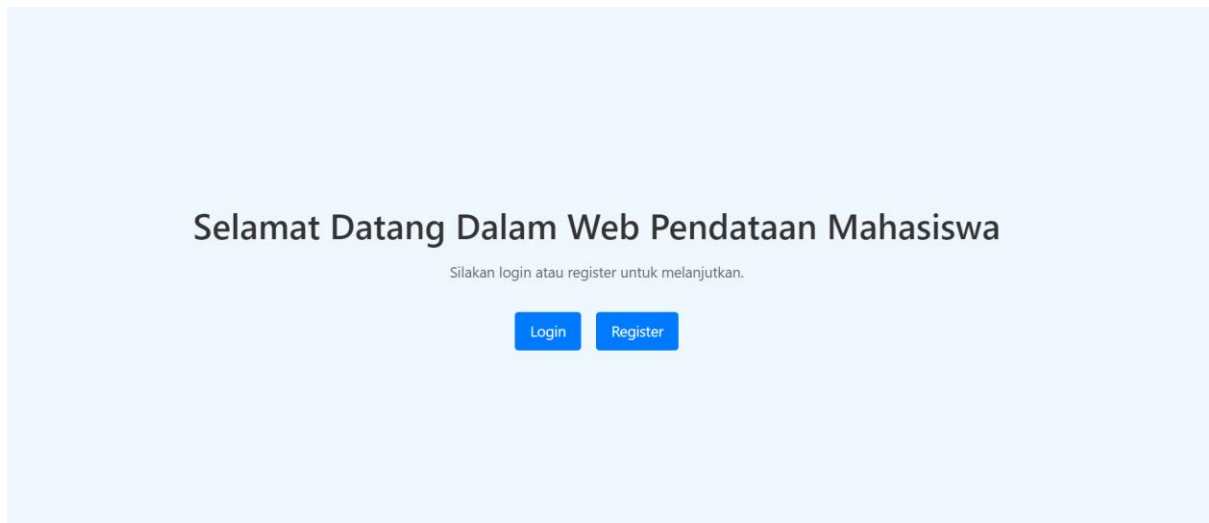
Register index

```
<body>
<div class="container">
  <h2 class="my-4" style="text-align:center">Register</h2>
  <div class="row justify-content-center">
    <div class="col-lg-5">
      <form action="/register" method="POST">
        <csrf>
        <div class="form-floating">
          <input type="text" name="name" class="form-control rounded-top @error('name') is-invalid @enderror" id="name" placeholder="Name" required value="{{ old('name') }}">
          <label for="name">Name</label>
          @error('name')
            <div class="invalid-feedback">
              {{ $message }}
            </div>
          @enderror
        </div>
        <div class="form-floating">
          <input type="text" name="username" class="form-control @error('username') is-invalid @enderror" id="username" placeholder="Username" required value="{{ old('username') }}">
          <label for="username">Username</label>
          @error('username')
            <div class="invalid-feedback">
              {{ $message }}
            </div>
          @enderror
        </div>
        <div class="form-floating">
          <input type="email" name="email" class="form-control @error('email') is-invalid @enderror" id="email" placeholder="name@example.com" required value="{{ old('email') }}">
          <label for="email">Email address</label>
          @error('email')
            <div class="invalid-feedback">
              {{ $message }}
            </div>
          @enderror
        </div>
        <div class="form-floating">
          <input type="password" name="password" class="form-control rounded-bottom @error('password') is-invalid @enderror" id="password" placeholder="Password" required>
          <label for="password">Password</label>
          @error('password')
            <div class="invalid-feedback">
              {{ $message }}
            </div>
          @enderror
        </div>
        <button class="btn btn-primary w-100 py-2 mt-3" type="submit">Register</button>
      </form>
      <small class="d-block text-center mt-3">Already Have An Account? <a href="/login">Login</a></small>
    </div>
  </div>
</div>
```

Gambar di atas merupakan code dari halaman register. terdapat div untuk membungkus konten halaman yang di dalamnya terdapat judul. Lalu diikuti, form registrasi. Di dalam Form registrasi terdapat atribut action="/register" untuk mengirimkan data ke URL /register dengan metode POST. Token CSRF disertakan menggunakan @csrf untuk keamanan. Form ini terdiri dari empat input yaitu, nama, username, email, dan password. Tombol submit digunakan untuk mengirimkan formulir, dan sebuah link untuk login pengguna yang sudah memiliki akun ditampilkan di bawah tombol submit, dengan teks "Already Have An Account? Login".

2. Screenshot Ouput

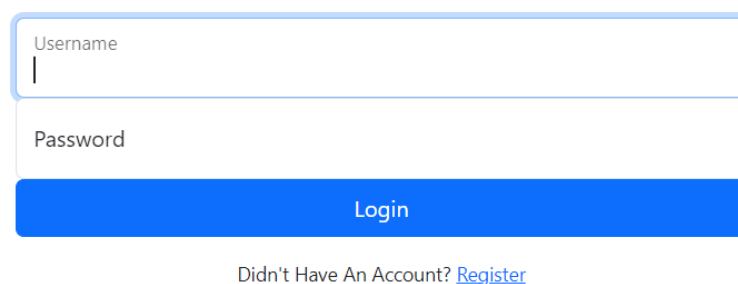
Home



Ini adalah halaman home. Halaman home ini sangat sederhana yang berisi tulisan selamat datang dan perintah untuk login untuk melanjutkan. Di bawahnya, terdapat button login dan juga register. Jika button ini di klik, maka program akan mengarahkannya ke halaman login atau register. Ini artinya semua user yang ingin melihat data dan melakukan CRUD harus melakukan login/register.

Login

Login

The screenshot shows a white rectangular form with a light blue border. Inside the form, there are two input fields. The first field is labeled 'Username' and has a blue border. The second field is labeled 'Password' and has a blue border. Below the input fields, there is a blue button with white text that says 'Login'. At the bottom of the form, there is a link that says 'Didn't Have An Account? [Register](#)'.

Ini adalah halaman login, halaman login terdiri dari username dan password. Pada bagian username terdapat autofocus yang membuat user langsung dapat mengisi field username secara langsung.

Register

Register

Register

Already Have An Account? [Login](#)

Ini adalah halaman register yang terdiri dari 4 inputan berupa nama, username, email, dan password. Pada bagian ini, user akan melakukan register agar masuk ke dalam database dan bisa masuk kehalaman utama.

Index

Data Mahasiswa						
No	Nama	NIM	Prodi	Kelas	Angkatan	Aksi
Tambah Data						Logout

Ini adalah halaman utama dari data mahasiswa. Halaman ini terdapat tabel yang menampilkan data nama, nim, prodi, kelas, angkatan, dan terdapat aksi untuk melakukan edit dan delete. Pada bagian bawah terdapat button untuk melakukan penambahan data dan juga melakukan logout. Jika button logut dipencet, maka akan masuk ke halaman awal yang berupa judul.

Student/Form

Form Data Mahasiswa

Nama	<input type="text" value="Masukkan nama anda"/>
NIM	<input type="text" value="Masukkan NIM anda"/>
Prodi	<input type="text" value="Masukkan prodi anda"/>
Kelas	<input type="text" value="Masukkan kelas anda"/>
Angkatan	<input type="text" value="Masukkan angkatan anda"/>
<input type="button" value="Tambah Data"/>	

Ini adalah halaman form tambah data. Halaman form ini dapat digunakan saat user memencet tombol tambah data. Di dalam form ini, terdapat baris-baris kolom yang bisa diisi oleh user. Data yang diisi berupa nama, nim, prodi, kelas, dan Angkatan. Pada bagian bawah terdapat button untuk tambah data.

Student/edit

Edit Form Data Mahasiswa

Nama	<input type="text" value="Afif Imam Rahadi"/>
NIM	<input type="text" value="L0122006"/>
Prodi	<input type="text" value="Informatika"/>
Kelas	<input type="text" value="A"/>
Angkatan	<input type="text" value="2022"/>
<input type="button" value="Edit Data"/>	

Ini adalah halaman form edit. Bentuk dari edit ini sama seperti form pengisian data. Pada saat user memencet tombol edit. Maka tampilan awal akan seperti di atas yaitu sesuai dengan data yang diinputkan. Jadi, jika user ingin mengubah salah satu data, user hanya tinggal menggantikan bagian data yang ingin diganti.

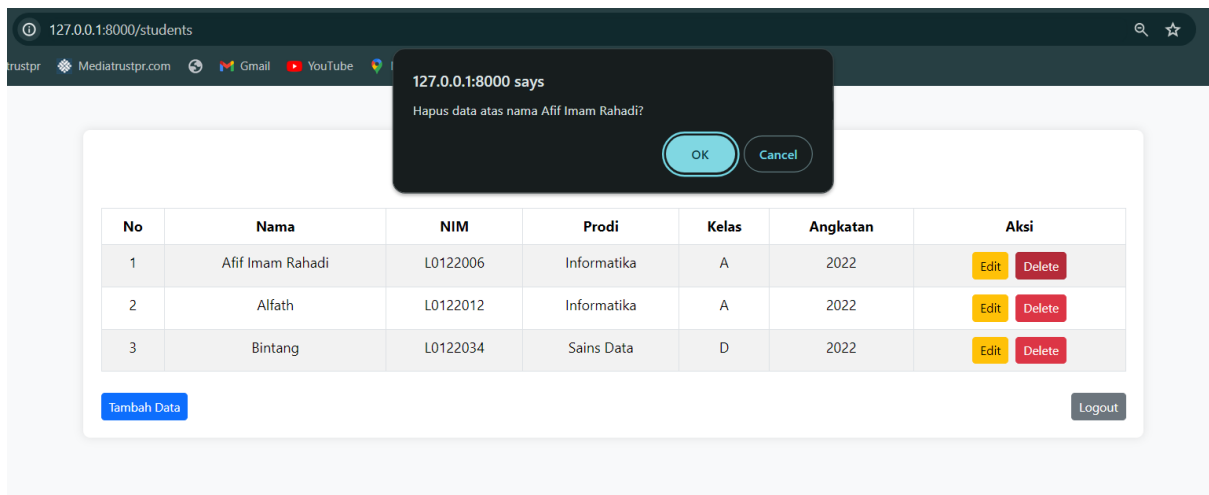
Tampilan setelah ditambah data

Data Mahasiswa						
No	Nama	NIM	Prodi	Kelas	Angkatan	Aksi
1	Afif Imam Rahadi	L0122006	Informatika	A	2022	<button>Edit</button> <button>Delete</button>
2	Alfath	L0122012	Informatika	A	2022	<button>Edit</button> <button>Delete</button>
3	Bintang	L0122034	Sains Data	D	2022	<button>Edit</button> <button>Delete</button>

Tambah Data Logout

Ini adalah tampilan setelah user menambahkan data. Tampilan berupa table yang berisi data-data yang diinputkan oleh user. Dari data ini, user dapat edit ataupun delete.

Delete Data



Ini adalah tampilan saat ingin menghapus data. Pada saat ingin menghapus, terdapat alert yang diikuti oleh pesan nama data yang ingin dihapus. Untuk menghapus, user hanya tinggal klik OK.