Nama : Afifah Nur Latifah

Rekrutmen Programmer Diskominfo

Soal A – Database

1. Membuat Tabel Users

- Membuat Database

Pertama, buat database "sekolahku":

CREATE DATABASE sekolahku;

USE sekolahku;

- Membuat Tabel "users"

Tabel "users" memiliki struktur/model seperti berikut:

```
CREATE TABLE users (
   id INT(11) AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
   email VARCHAR(50) NOT NULL,
   password VARCHAR(255) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

- Memasukkan Data ke Tabel "users"

Kemudian, masukkan beberapa data peserta didik ke dalam tabel "users".

Hasil:

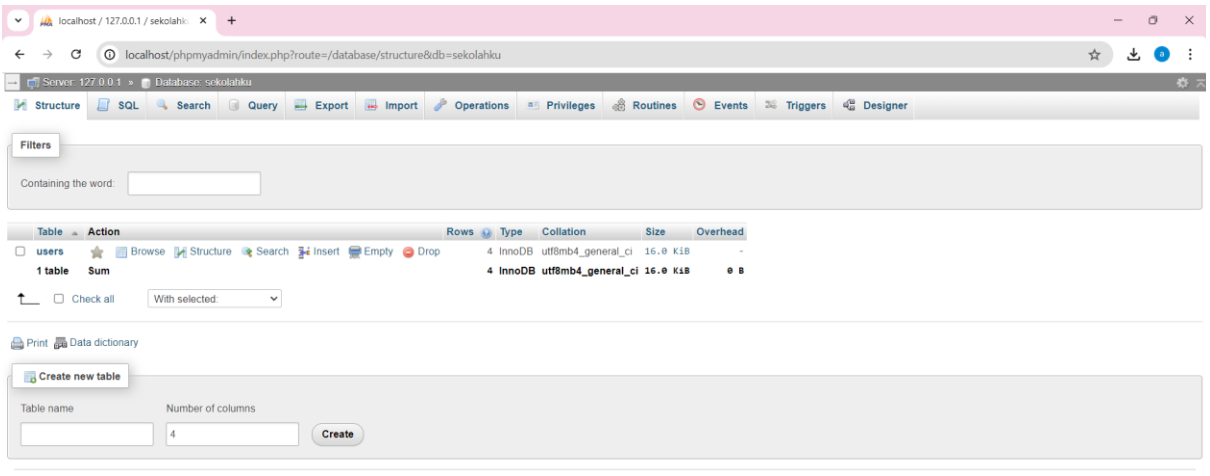
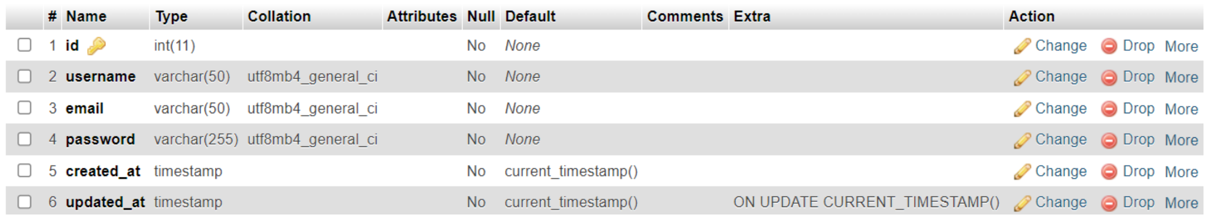
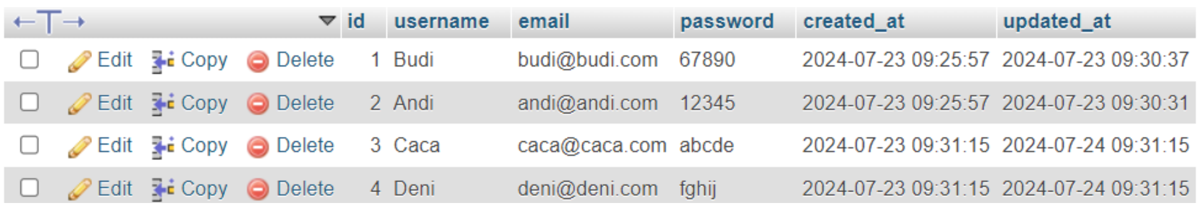

localhost / 127.0.0.1 / sekolahku
localhost/phpmyadmin/index.php?route=/database/structure&db=sekolahku
Server: 127.0.0.1 » Database: sekolahku
Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer
Filters
Containing the word:
Check all
With selected:
Print
Data dictionary
Create new table
Table name
Number of columns
4
Create

Table	Action	Rows	Type	Collation	Size	Overhead
users	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
1 table	Sum	4	InnoDB	utf8mb4_general_ci	16.0 KiB	0 B



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id**	int(11)			No	*None*			Change Drop More
2	**username**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
3	**email**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
4	**password**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
5	**created_at**	timestamp			No	current_timestamp()			Change Drop More
6	**updated_at**	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More



	id	username	email	password	created_at	updated_at
Edit Copy Delete	1	Budi	budi@budi.com	67890	2024-07-23 09:25:57	2024-07-23 09:30:37
Edit Copy Delete	2	Andi	andi@andi.com	12345	2024-07-23 09:25:57	2024-07-23 09:30:31
Edit Copy Delete	3	Caca	caca@caca.com	abcde	2024-07-23 09:31:15	2024-07-24 09:31:15
Edit Copy Delete	4	Deni	deni@deni.com	fghij	2024-07-23 09:31:15	2024-07-24 09:31:15

2. Membuat Tabel "courses"

Tabel "courses" dengan struktur sebagai berikut:

```
CREATE TABLE courses (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    course VARCHAR(50) NOT NULL,
    mentor VARCHAR(50) NOT NULL,
    title VARCHAR(50) NOT NULL
);
```

Hasil:

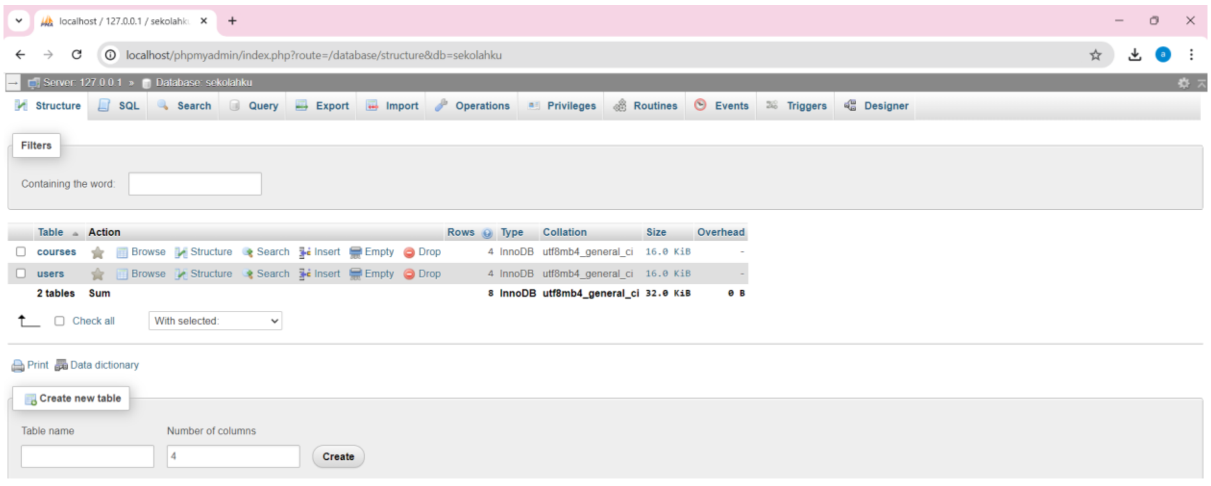
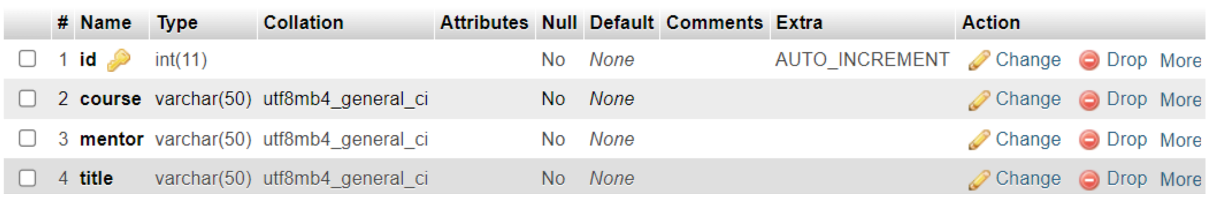
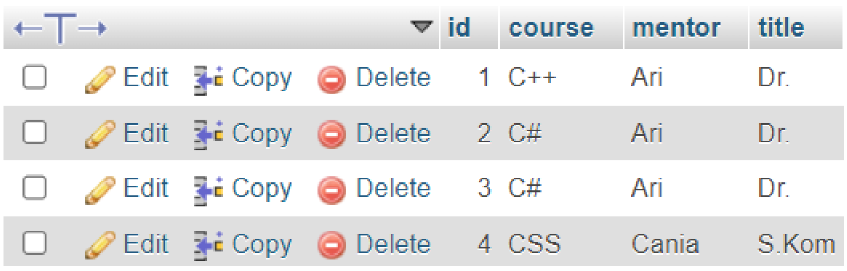

localhost / 127.0.0.1 / sekolahku
localhost/phpmyadmin/index.php?route=/database/structure&db=sekolahku
Server: 127.0.0.1 » Database: sekolahku
Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer
Filters
Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
courses	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
users	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
2 tables	Sum	8	InnoDB	utf8mb4_general_ci	32.0 KiB	0 B

Check all
With selected:
Print Data dictionary
Create new table
Table name
Number of columns
4
Create



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	**course**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
3	**mentor**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
4	**title**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More



	id	course	mentor	title
Edit Copy Delete	1	C++	Ari	Dr.
Edit Copy Delete	2	C#	Ari	Dr.
Edit Copy Delete	3	C#	Ari	Dr.
Edit Copy Delete	4	CSS	Cania	S.Kom

3. Membuat Tabel "userCourse"

Tabel "userCourse" untuk menyimpan data hubungan antara peserta didik dan kursus yang diikuti:

```
CREATE TABLE userCourse (
    user_id INT(11) NOT NULL,
    course_id INT(11) NOT NULL,
);
```

Hasil:

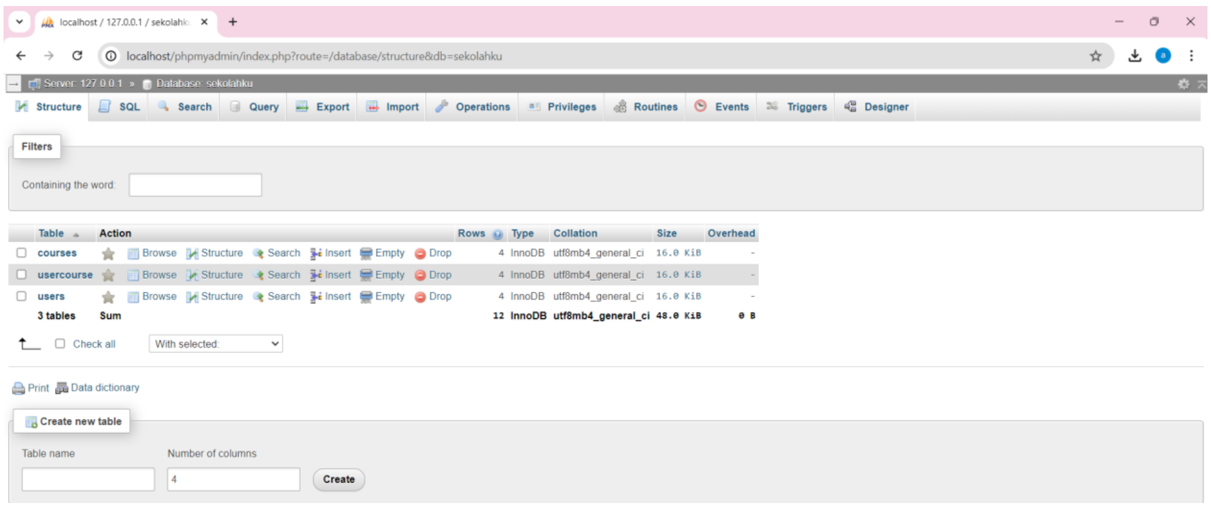
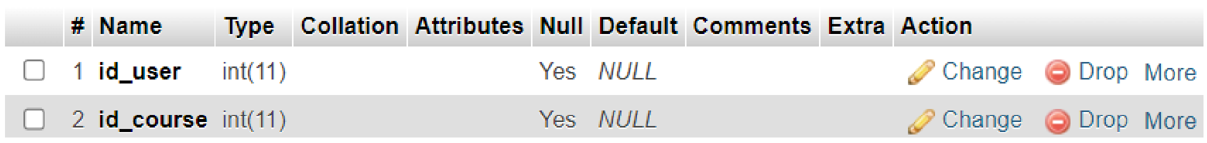
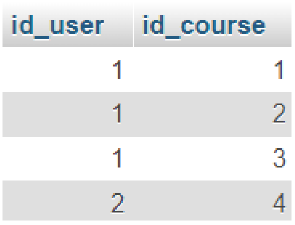

localhost / 127.0.0.1 / sekolahku
localhost/phpmyadmin/index.php?route=/database/structure&db=sekolahku
Server: 127.0.0.1 » Database: sekolahku
Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Designer
Filters
Containing the word:

	Table	Action	Rows	Type	Collation	Size	Overhead
	courses	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	usercourse	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	users	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	3 tables	Sum	12	InnoDB	utf8mb4_general_ci	48.0 KiB	0 B

Check all
With selected:
Print Data dictionary
Create new table
Table name
Number of columns
4
Create



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id_user**	int(11)			Yes	*NULL*			Change Drop More
2	**id_course**	int(11)			Yes	*NULL*			Change Drop More



id_user	id_course
1	1
1	2
1	3
2	4

4. Untuk menampilkan semua daftar peserta didik beserta mata kuliah yang diikutinya lengkap dengan nama dan gelar mentornya, kita perlu melakukan JOIN antara tabel "users", "courses", dan "userCourse". Berikut adalah query SQL nya:

```
SELECT
        users.id,
        users.username,
        courses.course,
        courses.mentor,
        courses.title
FROM
        userCourse usercourse
JOIN
        users users ON usercourse.user_id = users.id
JOIN
        courses courses ON usercourse_id = courses.id;
```

5. Untuk menampilkan daftar peserta didik beserta mata kuliah yang diikutinya, yang mentornya bergelar sarjana, kita dapat menambahkan kondisi pada query untuk memeriksa apakah mentor memiliki gelar sarjana. Misalkan kita mengasumsikan bahwa gelar sarjana ditandai dengan string "S" (misalnya, "S.Pd", "S.T", "S.Kom", dll.) di kolom mentor.

Berikut adalah query yang sesuai:

```
SELECT
    users.id,
    users.username,
    courses.course,
    courses.mentor,
    courses.title
FROM
    userCourse usercourse
JOIN
    users users ON usercourse.user_id = users.id
JOIN
```

```
    courses courses ON usercourse.course_id = courses.id
WHERE
    courses.mentor LIKE '%S%';
```

6. Untuk menampilkan daftar peserta didik beserta mata kuliah yang diikutinya dengan ketentuan bahwa mentornya bergelar selain sarjana (misalnya magister atau doktor), kita perlu memastikan bahwa kolom mentor di tabel courses tidak mengandung gelar sarjana dan mengandung gelar magister atau doktor.

Misalkan gelar sarjana diwakili dengan string "S" (misalnya "S.Pd", "S.T", dll.), dan gelar magister atau doktor diwakili dengan string "M" (misalnya "M.Pd", "M.T", "M.Kom", dll.) atau "Dr." (misalnya "Dr. Andi"). Berikut adalah query SQL yang sesuai:

```
SELECT
    users.id,
    users.username,
    courses.course,
    courses.mentor,
    courses.title
FROM
    userCourse usercourse
JOIN
    users users ON usercourse.user_id = users.id
JOIN
    courses courses ON usercourse.course_id = courses.id
WHERE
    courses.mentor NOT LIKE '%S%'
    AND (courses.mentor LIKE '%M%' OR courses.mentor LIKE 'Dr.%');
```

7. Untuk menampilkan jumlah peserta didik untuk setiap mata kuliah dari tabel users, courses, dan userCourse dapat menggunakan query SQL yang menggabungkan tabel-tabel tersebut dan menghitung jumlah peserta didik untuk setiap mata kuliah. Berikut adalah query SQL yang dapat digunakan.

```
SELECT
    courses.course,
    courses.mentor,
    courses.title,
    COUNT(usercourse.user_id) AS jumlah_peserta
FROM
    courses courses
LEFT JOIN
    userCourse usercourse ON courses.id = usercourse.course_id
GROUP BY
    courses.id, courses.course, courses.mentor, courses.title;
```

8. Untuk menampilkan jumlah peserta didik dan total fee untuk setiap mentor perlu menggabungkan data dari tabel users, courses, dan userCourse, serta menghitung jumlah peserta didik dan total fee per mentor. Berikut langkah-langkah dan query SQL yang dapat digunakan:

```
SELECT
    courses.mentor,
    COUNT(usercourse.user_id) AS jumlah_peserta,
    COUNT(usercourse.user_id) * 2000000 AS total_fee
FROM
    courses courses
LEFT JOIN
    userCourse usercourse ON courses.id = usercourse.course_id
GROUP BY
    courses.mentor;
```

Soal B - Live Coding

1. Setup Proyek Laravel

1. **Instal Laravel**:

```
composer create-project --prefer-dist laravel/laravel crudApp
```

2. **Masuk ke Direktori Proyek**:

```
cd crudApp
```

3. **Konfigurasi Database**: Edit file .env untuk menambahkan konfigurasi database:

```
dotenv
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=sekolahku
DB_USERNAME=root
DB_PASSWORD=
```

2. Buat Model, Controller, dan Migration

1. **Buat Model dan Migration**:

```
php artisan make:model User -m
php artisan make:model Course -m
php artisan make:model UserCourse -m
```

2. **Edit Migration Files**:

database/migrations/xxxx_xx_xx_create_users_table.php:

```
public function up()
{
   Schema::create('users', function (Blueprint $table) {
      $table->id();
      $table->string('username')->unique();
      $table->string('email')->unique();
      $table->string('password');
      $table->string('role'); // 'admin' or 'user'
      $table->timestamps();
   });
}
```

database/migrations/xxxx_xx_xx_create_courses_table.php:

```
public function up()
{
   Schema::create('courses', function (Blueprint $table) {
        $table->id();
        $table->string('course');
        $table->string('mentor');
        $table->string('title');
        $table->timestamps();
    });
}
```

database/migrations/xxxx_xx_xx_create_user_courses_table.php:

```
public function up()
{
    Schema::create('user_courses', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained('users')->onDelete('cascade');
        $table->foreignId('course_id')->constrained('courses')->onDelete('cascade');
        $table->timestamps();
    });
}
```

3. **Jalankan Migrasi**:

```
php artisan migrate
```

3. Buat Controller dan Route

1. **Buat Controller**:

```
php artisan make:controller UserController
php artisan make:controller CourseController
php artisan make:controller UserCourseController
php artisan make:controller DashboardController
php artisan make:controller ApiController
```

2. **Edit Controller**:

app/Http/Controllers/UserController.php:

```
<?php
```

```
namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class UserController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    public function index()
    {
        if (Auth::user()->role == 'admin') {
            $users = User::all();
        } else {
            $users = User::where('id', Auth::id())->get();
        }
        return view('users.index', compact('users'));
    }

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

    public function store(Request $request)
    {
        $request->validate([
            'username' => 'required|unique:users',
            'email' => 'required|email|unique:users',
            'password' => 'required',
            'role' => 'required|in:admin,user'
        ]);

        User::create([
            'username' => $request->username,
            'email' => $request->email,
            'password' => bcrypt($request->password),
            'role' => $request->role,
        ]);

        return redirect()->route('users.index');
    }

    public function edit(User $user)
    {
        return view('users.edit', compact('user'));
    }

    public function update(Request $request, User $user)
    {
        $request->validate([
            'username' => 'required|unique:users,username,' . $user->id,
            'email' => 'required|email|unique:users,email,' . $user->id,
            'password' => 'nullable',
            'role' => 'required|in:admin,user'
        ]);
        $user->update([
            'username' => $request->username,
            'email' => $request->email,
            'password' => $request->password ? bcrypt($request->password) : $user->password,
            'role' => $request->role,
        ]);

        return redirect()->route('users.index');
    }

    public function destroy(User $user)
    {
        $user->delete();
        return redirect()->route('users.index');
    }
}
```

app/Http/Controllers/CourseController.php dan **app/Http/Controllers/UserCourseController.php** mengikuti pola yang sama dengan UserController.php.

3. **Edit Routes**:

routes/web.php:

```
use App\Http\Controllers\UserController;
use App\Http\Controllers\CourseController;
use App\Http\Controllers\UserCourseController;
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\ApiController;

Route::middleware(['auth'])->group(function () {
    Route::resource('users', UserController::class);
    Route::resource('courses', CourseController::class);
    Route::resource('userCourses', UserCourseController::class);
    Route::get('dashboard', [DashboardController::class, 'index'])->name('dashboard');
});

// API Routes
Route::prefix('api')->group(function () {
    Route::get('users', [ApiController::class, 'users']);
    Route::get('courses', [ApiController::class, 'courses']);
    Route::get('userCourses', [ApiController::class, 'userCourses']);
});
```

4. Buat Grafik Dashboard Admin

app/Http/Controllers/DashboardController.php:

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Models\Course;
use App\Models\UserCourse;

class DashboardController extends Controller
{
    public function index()
    {
        $userCount = User::count();
        $courseCount = Course::count();
        $userCourseCount = UserCourse::count();

        return view('dashboard.index', compact('userCount', 'courseCount', 'userCourseCount'));
    }
}
```

resources/views/dashboard/index.blade.php:

```
<!DOCTYPE html>
<html>
<head>
  <title>Admin Dashboard</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h1>Admin Dashboard</h1>
  <canvas id="dashboardChart" width="400" height="200"></canvas>
  <script>
  var ctx = document.getElementById('dashboardChart').getContext('2d');
  var chart = new Chart(ctx, {
    type: 'bar',
    data: {
      labels: ['Users', 'Courses', 'User Courses'],
      datasets: [{
        label: 'Counts',
        data: [{{ $userCount }}, {{ $courseCount }}, {{ $userCourseCount }}],
        backgroundColor: 'rgba(54, 162, 235, 0.2)',
        borderColor: 'rgba(54, 162, 235, 1)',
        borderWidth: 1
      }]
    },
    options: {
      scales: {
        y: {
          beginAtZero: true
        }
```

```
            }
        }
    });
    </script>
</body>
</html>
```

5. Buat API Endpoint

app/Http/Controllers/ApiController.php:

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Models\Course;
use App\Models\UserCourse;

class ApiController extends Controller
{
    public function users()
    {
        return response()->json(User::all());
    }

    public function courses()
    {
        return response()->json(Course::all());
    }

    public function userCourses()
    {
        $userCourses = UserCourse::with(['user', 'course'])->get();
        return response()->json($userCourses);
    }
}
```

6. Buat Unit Test

1. **Instal PHPUnit** (Jika belum ada):

```
composer require --dev phpunit/phpunit
```

2. **Buat Unit Test**:

tests/Feature/LoginTest.php:

```
<?php

namespace Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Support\Facades\Hash;
use App\Models\User;

class LoginTest extends TestCase
{
    use RefreshDatabase;

    public function testValidLogin()
    {
        $user = User::factory()->create([
            'password' => Hash::make('password')
        ]);

        $response = $this->post('/login', [
            'username' => $user->username,
            'password' => 'password'
        ]);
```