Tutorial Lengkap Pengembangan Backend API E-Learning BPSDM dengan Laravel dan Laravel Module

Berikut adalah tutorial pengembangan backend API untuk sistem e-learning BPSDM menggunakan Laravel dan Laravel Module berdasarkan ERD yang diberikan. Tutorial ini berfokus hanya pada pengembangan RESTful API tanpa tampilan frontend.

Daftar Isi

1. Persiapan Awal

Instalasi Laravel

bash

```
composer create-project laravel/laravel bpsdm-elearning-api
cd bpsdm-elearning-api
```

Instalasi Laravel Module

bash

```
composer require nwidart/laravel-modules
```

Publish Konfigurasi Laravel Module

bash

```
php artisan vendor:publish --provider="Nwidart\Modules\LaravelModulesServiceProvider"
```

```
```

Konfigurasi Database di .env

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

Instalasi Laravel Sanctum untuk API Authentication

bash

```
composer require laravel/sanctum
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
php artisan migrate
```

Konfigurasi CORS di config/cors.php

php

```
return [
    'paths' => ['api/*', 'sanctum/csrf-cookie'],
    'allowed_methods' => ['*'],
    'allowed_origins' => ['*'],
    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => true,
];
```

2. Struktur Modul

Berdasarkan ERD, kita akan membuat struktur modul sebagai berikut:

bash

```
php artisan module:make Auth
php artisan module:make Master
php artisan module:make Kursus
php artisan module:make Materi
php artisan module:make Evaluasi
php artisan module:make Kehadiran
```

```
php artisan module:make Kelompok
php artisan module:make Forum
php artisan module:make Sertifikat
php artisan module:make Laporan
```

3. Konfigurasi API Authentication

Konfigurasi Sanctum

Edit file `app/Http/Kernel.php`:

php

```php
protected $middlewareGroups = [
    // ...
    'api' => [
        \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
        'throttle:api',
        \Illuminate\Routing\Middleware\SubstituteBindings::class,
    ],
];
```

Konfigurasi Auth

php

```php
// config/auth.php
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'admin' => [
        'driver' => 'sanctum',
        'provider' => 'admin',
    ],
    'peserta' => [
        'driver' => 'sanctum',
        'provider' => 'peserta',
    ],
],

'providers' => [
```

```
    'users' => [
        'driver' => 'eloquent',
        'model' => App\Models\User::class,
    ],
    'admin' => [
        'driver' => 'eloquent',
        'model' => Modules\Auth\Entities\AdminInstruktur::class,
    ],
    'peserta' => [
        'driver' => 'eloquent',
        'model' => Modules\Auth\Entities\Peserta::class,
    ],
],
```

4. Pengembangan Modul Auth

Migrasi Database

bash

```
php artisan module:make-migration create_admin_instruktur_table --module=Auth
php artisan module:make-migration create_peserta_table --module=Auth
```

File migrasi admin_instruktur:

php

```
// Modules/Auth/Database/Migrations/xxxx_xx_xx_create_admin_instruktur_table.php
public function up()
{
    Schema::create('admin_instruktur', function (Blueprint $table) {
        $table->id();
        $table->string('username')->unique();
        $table->string('email')->unique();
        $table->string('password');
        $table->enum('role', ['super_admin', 'instruktur']);
        $table->string('nama_lengkap');
        $table->string('nip')->nullable();
        $table->string('gelar_depan')->nullable();
        $table->string('gelar_belakang')->nullable();
        $table->text('bidang_keahlian')->nullable();
        $table->string('no_telepon')->nullable();
        $table->text('alamat')->nullable();
```

```
            $table->string('foto_profil')->nullable();
            $table->timestamp('email_verified_at')->nullable();
            $table->rememberToken();
            $table->timestamps();
            $table->softDeletes();
        });
    }
```

File migrasi peserta:

php

```
// Modules/Auth/Database/Migrations/xxxx_xx_xx_create_peserta_table.php
public function up()
{
    Schema::create('peserta', function (Blueprint $table) {
        $table->id();
        $table->unsignedBigInteger('opd_id');
        $table->string('username')->unique();
        $table->string('email')->unique();
        $table->string('password');
        $table->string('nama_lengkap');
        $table->string('nip')->nullable();
        $table->string('pangkat_golongan')->nullable();
        $table->string('jabatan')->nullable();
        $table->date('tanggal_lahir')->nullable();
        $table->string('tempat_lahir')->nullable();
        $table->enum('jenis_kelamin', ['laki_laki', 'perempuan'])->nullable();
        $table->enum('pendidikan_terakhir', ['sma', 'd3', 's1', 's2', 's3'])->nullable();
        $table->enum('status_kepegawaian', ['pns', 'pppk', 'kontrak'])->nullable();
        $table->string('no_telepon')->nullable();
        $table->text('alamat')->nullable();
        $table->string('foto_profil')->nullable();
        $table->timestamp('email_verified_at')->nullable();
        $table->rememberToken();
        $table->timestamps();
        $table->softDeletes();

        $table->foreign('opd_id')->references('id')->on('opd');
    });
}
```

Models

bash

```
php artisan module:make-model AdminInstruktur --module=Auth
php artisan module:make-model Peserta --module=Auth
```

Model AdminInstruktur:

php

```
// Modules/Auth/Entities/AdminInstruktur.php
namespace Modules\Auth\Entities;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Sanctum\HasApiTokens;
use Modules\Kursus\Entities\Kursus;
use Modules\Evaluasi\Entities\PenilaianTugas;

class AdminInstruktur extends Authenticatable
{
    use HasApiTokens, SoftDeletes;

    protected $table = 'admin_instruktur';

    protected $fillable = [
        'username', 'email', 'password', 'role', 'nama_lengkap',
        'nip', 'gelar_depan', 'gelar_belakang', 'bidang_keahlian',
        'no_telepon', 'alamat', 'foto_profil'
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    public function kursus()
    {
        return $this->hasMany(Kursus::class, 'admin_instruktur_id');
    }
```

```
    public function penilaianTugas()
    {
        return $this->hasMany(PenilaianTugas::class, 'admin_instruktur_id');
    }
}
```

Model Peserta:

php

```
// Modules/Auth/Entities/Peserta.php
namespace Modules\Auth\Entities;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Sanctum\HasApiTokens;
use Modules\Master\Entities\OPD;
use Modules\Kursus\Entities\PendaftaranKursus;

class Peserta extends Authenticatable
{
    use HasApiTokens, SoftDeletes;

    protected $table = 'peserta';

    protected $fillable = [
        'opd_id', 'username', 'email', 'password', 'nama_lengkap',
        'nip', 'pangkat_golongan', 'jabatan', 'tanggal_lahir',
        'tempat_lahir', 'jenis_kelamin', 'pendidikan_terakhir',
        'status_kepegawaian', 'no_telepon', 'alamat', 'foto_profil'
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    public function opd()
    {
        return $this->belongsTo(OPD::class, 'opd_id');
    }
```

```
    public function pendaftaranKursus()
    {
        return $this->hasMany(PendaftaranKursus::class, 'peserta_id');
    }
}
```

API Controllers

bash

```
php artisan module:make-controller AuthController --module=Auth
```

AuthController:

php

```
// Modules/Auth/Http/Controllers/AuthController.php
namespace Modules\Auth\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\ValidationException;
use Modules\Auth\Entities\AdminInstruktur;
use Modules\Auth\Entities\Peserta;

class AuthController extends Controller
{
    public function loginAdmin(Request $request)
    {
        $request->validate([
            'username' => 'required',
            'password' => 'required',
        ]);

        $admin = AdminInstruktur::where('username', $request->username)
                    ->orWhere('email', $request->username)
                    ->first();

        if (!$admin || !Hash::check($request->password, $admin->password)) {
            throw ValidationException::withMessages([
```

```
            'username' => ['Kredensial yang diberikan tidak cocok dengan catatan kami.'],
        ]);
    }

    return response()->json([
        'status' => 'success',
        'message' => 'Login berhasil',
        'data' => [
            'user' => $admin,
            'token' => $admin->createToken('admin-token')->plainTextToken,
            'token_type' => 'Bearer'
        ]
    ]);
}

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

    $peserta = Peserta::where('username', $request->username)
                ->orWhere('email', $request->username)
                ->first();

    if (!$peserta || !Hash::check($request->password, $peserta->password)) {
        throw ValidationException::withMessages([
            'username' => ['Kredensial yang diberikan tidak cocok dengan catatan kami.'],
        ]);
    }

    return response()->json([
        'status' => 'success',
        'message' => 'Login berhasil',
        'data' => [
            'user' => $peserta,
            'token' => $peserta->createToken('peserta-token')->plainTextToken,
            'token_type' => 'Bearer'
```

```
            ]
        ]);
    }

    public function logout(Request $request)
    {
        $request->user()->currentAccessToken()->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Logout berhasil'
        ]);
    }

    public function profile(Request $request)
    {
        return response()->json([
            'status' => 'success',
            'data' => $request->user()
        ]);
    }

    public function updateProfile(Request $request)
    {
        $user = $request->user();

        $rules = [
            'nama_lengkap' => 'required|string|max:100',
            'email' => 'required|email|unique:' . ($user instanceof AdminInstruktur ? 'admin_instruktur' :
'peserta') . ',email,' . $user->id,
            'no_telepon' => 'nullable|string|max:20',
            'alamat' => 'nullable|string',
        ];

        if ($user instanceof Peserta) {
            $rules['pangkat_golongan'] = 'nullable|string|max:50';
            $rules['jabatan'] = 'nullable|string|max:100';
            $rules['pendidikan_terakhir'] = 'nullable|in:sma,d3,s1,s2,s3';
        }
```

```
    if ($user instanceof AdminInstruktur) {
        $rules['gelar_depan'] = 'nullable|string|max:20';
        $rules['gelar_belakang'] = 'nullable|string|max:50';
        $rules['bidang_keahlian'] = 'nullable|string';
    }

    $request->validate($rules);

    $user->update($request->only(array_keys($rules)));

    return response()->json([
        'status' => 'success',
        'message' => 'Profil berhasil diperbarui',
        'data' => $user
    ]);
}

public function changePassword(Request $request)
{
    $request->validate([
        'current_password' => 'required',
        'password' => 'required|min:6|confirmed',
    ]);

    $user = $request->user();

    if (!Hash::check($request->current_password, $user->password)) {
        throw ValidationException::withMessages([
            'current_password' => ['Password saat ini tidak cocok.'],
        ]);
    }

    $user->update([
        'password' => Hash::make($request->password),
    ]);

    return response()->json([
        'status' => 'success',
        'message' => 'Password berhasil diubah'
```

```
        ]);
    }
}
```

Routes

php

```
// Modules/Auth/Routes/api.php
use Illuminate\Support\Facades\Route;
use Modules\Auth\Http\Controllers\AuthController;

Route::prefix('auth')->group(function () {
    Route::post('login/admin', [AuthController::class, 'loginAdmin']);
    Route::post('login/peserta', [AuthController::class, 'loginPeserta']);

    Route::middleware('auth:sanctum')->group(function () {
        Route::post('logout', [AuthController::class, 'logout']);
        Route::get('profile', [AuthController::class, 'profile']);
        Route::put('profile', [AuthController::class, 'updateProfile']);
        Route::post('change-password', [AuthController::class, 'changePassword']);
    });
});
```

5. Pengembangan Modul Master

Migrasi Database

bash

```
php artisan module:make-migration create_opd_table --module=Master
php artisan module:make-migration create_kategori_kursus_table --module=Master
```

File migrasi opd:

php

```
// Modules/Master/Database/Migrations/xxxx_xx_xx_create_opd_table.php
public function up()
{
    Schema::create('opd', function (Blueprint $table) {
        $table->id();
        $table->string('kode_opd')->unique();
        $table->string('nama_opd');
        $table->text('alamat')->nullable();
```

```
            $table->string('no_telepon')->nullable();
            $table->string('email')->nullable();
            $table->string('nama_kepala')->nullable();
            $table->timestamps();
        });
    }
```

File migrasi kategori_kursus:

php

```
// Modules/Master/Database/Migrations/xxxx_xx_xx_create_kategori_kursus_table.php
public function up()
{
    Schema::create('kategori_kursus', function (Blueprint $table) {
        $table->id();
        $table->string('nama_kategori');
        $table->string('slug')->unique();
        $table->text('deskripsi')->nullable();
        $table->string('icon')->nullable();
        $table->integer('urutan')->default(0);
        $table->timestamps();
    });
}
```

Models

bash

```
php artisan module:make-model OPD --module=Master
php artisan module:make-model KategoriKursus --module=Master
```

Model OPD:

php

```
// Modules/Master/Entities/OPD.php
namespace Modules\Master\Entities;

use Illuminate\Database\Eloquent\Model;
use Modules\Auth\Entities\Peserta;

class OPD extends Model
{
    protected $table = 'opd';
```

```
    protected $fillable = [
        'kode_opd', 'nama_opd', 'alamat', 'no_telepon', 'email', 'nama_kepala'
    ];

    public function peserta()
    {
        return $this->hasMany(Peserta::class, 'opd_id');
    }
}
```

Model KategoriKursus:

php

```
// Modules/Master/Entities/KategoriKursus.php
namespace Modules\Master\Entities;

use Illuminate\Database\Eloquent\Model;
use Modules\Kursus\Entities\Kursus;
use Illuminate\Support\Str;

class KategoriKursus extends Model
{
    protected $table = 'kategori_kursus';

    protected $fillable = [
        'nama_kategori', 'slug', 'deskripsi', 'icon', 'urutan'
    ];

    protected static function boot()
    {
        parent::boot();

        static::creating(function ($kategori) {
            $kategori->slug = $kategori->slug ?: Str::slug($kategori->nama_kategori);
        });

        static::updating(function ($kategori) {
            if ($kategori->isDirty('nama_kategori') && !$kategori->isDirty('slug')) {
                $kategori->slug = Str::slug($kategori->nama_kategori);
```

```
        }
    });
  }

  public function kursus()
  {
    return $this->hasMany(Kursus::class, 'kategori_id');
  }
}
```

API Controllers

bash

```
php artisan module:make-controller OPDController --module=Master
php artisan module:make-controller KategoriKursusController --module=Master
```

OPDController:

php

```
// Modules/Master/Http/Controllers/OPDController.php
namespace Modules\Master\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Master\Entities\OPD;
use Illuminate\Support\Facades\Validator;

class OPDController extends Controller
{
  public function index()
  {
    $opd = OPD::all();

    return response()->json([
      'status' => 'success',
      'data' => $opd
    ]);
  }

  public function store(Request $request)
  {
```

```
    $validator = Validator::make($request->all(), [
        'kode_opd' => 'required|unique:opd,kode_opd',
        'nama_opd' => 'required|string|max:100',
        'alamat' => 'nullable|string',
        'no_telepon' => 'nullable|string|max:20',
        'email' => 'nullable|email|max:100',
        'nama_kepala' => 'nullable|string|max:100',
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    $opd = OPD::create($request->all());

    return response()->json([
        'status' => 'success',
        'message' => 'OPD berhasil ditambahkan',
        'data' => $opd
    ], 201);
}

public function show($id)
{
    $opd = OPD::find($id);

    if (!$opd) {
        return response()->json([
            'status' => 'error',
            'message' => 'OPD tidak ditemukan'
        ], 404);
    }

    return response()->json([
        'status' => 'success',
```

```
            'data' => $opd
        ]);
    }

    public function update(Request $request, $id)
    {
        $opd = OPD::find($id);

        if (!$opd) {
            return response()->json([
                'status' => 'error',
                'message' => 'OPD tidak ditemukan'
            ], 404);
        }

        $validator = Validator::make($request->all(), [
            'kode_opd' => 'required|unique:opd,kode_opd,' . $id,
            'nama_opd' => 'required|string|max:100',
            'alamat' => 'nullable|string',
            'no_telepon' => 'nullable|string|max:20',
            'email' => 'nullable|email|max:100',
            'nama_kepala' => 'nullable|string|max:100',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
                'errors' => $validator->errors()
            ], 422);
        }

        $opd->update($request->all());

        return response()->json([
            'status' => 'success',
            'message' => 'OPD berhasil diperbarui',
            'data' => $opd
        ]);
```

```
    }

    public function destroy($id)
    {
        $opd = OPD::find($id);

        if (!$opd) {
            return response()->json([
                'status' => 'error',
                'message' => 'OPD tidak ditemukan'
            ], 404);
        }

        // Check if OPD has peserta
        $pesertaCount = $opd->peserta()->count();
        if ($pesertaCount > 0) {
            return response()->json([
                'status' => 'error',
                'message' => 'OPD tidak dapat dihapus karena masih memiliki ' . $pesertaCount . ' peserta'
            ], 400);
        }

        $opd->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'OPD berhasil dihapus'
        ]);
    }
}
```

KategoriKursusController:

php

```
// Modules/Master/Http/Controllers/KategoriKursusController.php
namespace Modules\Master\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Master\Entities\KategoriKursus;
```

```php
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;

class KategoriKursusController extends Controller
{
    public function index()
    {
        $kategori = KategoriKursus::orderBy('urutan', 'asc')->get();

        return response()->json([
            'status' => 'success',
            'data' => $kategori
        ]);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nama_kategori' => 'required|string|max:100',
            'slug' => 'nullable|string|unique:kategori_kursus,slug',
            'deskripsi' => 'nullable|string',
            'icon' => 'nullable|string|max:50',
            'urutan' => 'nullable|integer|min:0',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
                'errors' => $validator->errors()
            ], 422);
        }

        $data = $request->all();

        if (empty($data['slug'])) {
            $data['slug'] = Str::slug($data['nama_kategori']);
        }
```

```
    $kategori = KategoriKursus::create($data);

    return response()->json([
        'status' => 'success',
        'message' => 'Kategori kursus berhasil ditambahkan',
        'data' => $kategori
    ], 201);
}

public function show($id)
{
    $kategori = KategoriKursus::find($id);

    if (!$kategori) {
        return response()->json([
            'status' => 'error',
            'message' => 'Kategori kursus tidak ditemukan'
        ], 404);
    }

    return response()->json([
        'status' => 'success',
        'data' => $kategori
    ]);
}

public function update(Request $request, $id)
{
    $kategori = KategoriKursus::find($id);

    if (!$kategori) {
        return response()->json([
            'status' => 'error',
            'message' => 'Kategori kursus tidak ditemukan'
        ], 404);
    }

    $validator = Validator::make($request->all(), [
        'nama_kategori' => 'required|string|max:100',
```

```
            'slug' => 'nullable|string|unique:kategori_kursus,slug,' . $id,
            'deskripsi' => 'nullable|string',
            'icon' => 'nullable|string|max:50',
            'urutan' => 'nullable|integer|min:0',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
                'errors' => $validator->errors()
            ], 422);
        }

        $data = $request->all();

        if (empty($data['slug']) && $request->has('nama_kategori')) {
            $data['slug'] = Str::slug($data['nama_kategori']);
        }

        $kategori->update($data);

        return response()->json([
            'status' => 'success',
            'message' => 'Kategori kursus berhasil diperbarui',
            'data' => $kategori
        ]);
    }

    public function destroy($id)
    {
        $kategori = KategoriKursus::find($id);

        if (!$kategori) {
            return response()->json([
                'status' => 'error',
                'message' => 'Kategori kursus tidak ditemukan'
            ], 404);
        }
```

```
        // Check if kategori has kursus
        $kursusCount = $kategori->kursus()->count();
        if ($kursusCount > 0) {
            return response()->json([
                'status' => 'error',
                'message' => 'Kategori tidak dapat dihapus karena masih memiliki ' . $kursusCount . ' kursus'
            ], 400);
        }

        $kategori->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Kategori kursus berhasil dihapus'
        ]);
    }
}
```

Routes

php

```
// Modules/Master/Routes/api.php
use Illuminate\Support\Facades\Route;
use Modules\Master\Http\Controllers\OPDController;
use Modules\Master\Http\Controllers\KategoriKursusController;

Route::middleware('auth:sanctum')->group(function () {
    // OPD Routes
    Route::get('opd', [OPDController::class, 'index']);
    Route::post('opd', [OPDController::class, 'store'])->middleware('ability:super_admin');
    Route::get('opd/{id}', [OPDController::class, 'show']);
    Route::put('opd/{id}', [OPDController::class, 'update'])->middleware('ability:super_admin');
    Route::delete('opd/{id}', [OPDController::class, 'destroy'])->middleware('ability:super_admin');

    // Kategori Kursus Routes
    Route::get('kategori-kursus', [KategoriKursusController::class, 'index']);
    Route::post('kategori-kursus', [KategoriKursusController::class, 'store'])-
>middleware('ability:super_admin');
    Route::get('kategori-kursus/{id}', [KategoriKursusController::class, 'show']);
```

```
    Route::put('kategori-kursus/{id}', [KategoriKursusController::class, 'update'])->middleware('ability:super_admin');
    Route::delete('kategori-kursus/{id}', [KategoriKursusController::class, 'destroy'])->middleware('ability:super_admin');
});
```

6. Pengembangan Modul Kursus

Migrasi Database

bash

```
php artisan module:make-migration create_kursus_table --module=Kursus
php artisan module:make-migration create_prasyarat_table --module=Kursus
php artisan module:make-migration create_pendaftaran_kursus_table --module=Kursus
```

File migrasi kursus:

php

```
// Modules/Kursus/Database/Migrations/xxxx_xx_xx_create_kursus_table.php
public function up()
{
    Schema::create('kursus', function (Blueprint $table) {
        $table->id();
        $table->unsignedBigInteger('admin_instruktur_id');
        $table->unsignedBigInteger('kategori_id');
        $table->string('kode_kursus')->unique();
        $table->string('judul');
        $table->text('deskripsi');
        $table->text('tujuan_pembelajaran')->nullable();
        $table->text('sasaran_peserta')->nullable();
        $table->integer('durasi_jam')->default(0);
        $table->date('tanggal_buka_pendaftaran')->nullable();
        $table->date('tanggal_tutup_pendaftaran')->nullable();
        $table->date('tanggal_mulai_kursus')->nullable();
        $table->date('tanggal_selesai_kursus')->nullable();
        $table->integer('kuota_peserta')->default(0);
        $table->enum('level', ['dasar', 'menengah', 'lanjut'])->default('dasar');
        $table->enum('tipe', ['daring', 'luring', 'hybrid'])->default('daring');
        $table->enum('status', ['draft', 'aktif', 'nonaktif', 'selesai'])->default('draft');
        $table->string('thumbnail')->nullable();
        $table->decimal('passing_grade', 5, 2)->default(70.00);
        $table->timestamps();
```

```
            $table->foreign('admin_instruktur_id')->references('id')->on('admin_instruktur');
            $table->foreign('kategori_id')->references('id')->on('kategori_kursus');
        });
    }
```

Models

bash

```
php artisan module:make-model Kursus --module=Kursus
php artisan module:make-model Prasyarat --module=Kursus
php artisan module:make-model PendaftaranKursus --module=Kursus
```

Model Kursus:

php

```
// Modules/Kursus/Entities/Kursus.php
namespace Modules\Kursus\Entities;

use Illuminate\Database\Eloquent\Model;
use Modules\Auth\Entities\AdminInstruktur;
use Modules\Master\Entities\KategoriKursus;
use Modules\Materi\Entities\Modul;

class Kursus extends Model
{
    protected $table = 'kursus';

    protected $fillable = [
        'admin_instruktur_id', 'kategori_id', 'kode_kursus', 'judul',
        'deskripsi', 'tujuan_pembelajaran', 'sasaran_peserta',
        'durasi_jam', 'tanggal_buka_pendaftaran', 'tanggal_tutup_pendaftaran',
        'tanggal_mulai_kursus', 'tanggal_selesai_kursus', 'kuota_peserta',
        'level', 'tipe', 'status', 'thumbnail', 'passing_grade'
    ];

    protected $casts = [
        'tanggal_buka_pendaftaran' => 'date',
        'tanggal_tutup_pendaftaran' => 'date',
        'tanggal_mulai_kursus' => 'date',
        'tanggal_selesai_kursus' => 'date',
```

```
    'passing_grade' => 'float'
];

public function instruktur()
{
    return $this->belongsTo(AdminInstruktur::class, 'admin_instruktur_id');
}

public function kategori()
{
    return $this->belongsTo(KategoriKursus::class, 'kategori_id');
}

public function modul()
{
    return $this->hasMany(Modul::class, 'kursus_id');
}

public function pendaftaran()
{
    return $this->hasMany(PendaftaranKursus::class, 'kursus_id');
}

public function prasyarat()
{
    return $this->hasMany(Prasyarat::class, 'kursus_id');
}

public function getJumlahPesertaAttribute()
{
    return $this->pendaftaran()
        ->whereIn('status', ['disetujui', 'aktif', 'selesai'])
        ->count();
}

public function getProgressPersenAttribute()
{
    if (!$this->tanggal_mulai_kursus || !$this->tanggal_selesai_kursus) {
        return 0;
```

```
        }

        $now = now();
        $start = $this->tanggal_mulai_kursus;
        $end = $this->tanggal_selesai_kursus;

        if ($now < $start) {
            return 0;
        }

        if ($now > $end) {
            return 100;
        }

        $totalDays = $start->diffInDays($end) + 1;
        $passedDays = $start->diffInDays($now) + 1;

        return min(100, round(($passedDays / $totalDays) * 100));
    }
}
```

API Controllers

bash

```
php artisan module:make-controller KursusController --module=Kursus
php artisan module:make-controller PendaftaranController --module=Kursus
```

KursusController:

php

```
// Modules/Kursus/Http/Controllers/KursusController.php
namespace Modules\Kursus\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Kursus\Entities\Kursus;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
```

```
class KursusController extends Controller
{
    public function index(Request $request)
    {
        $query = Kursus::with(['kategori:id,nama_kategori',
'instruktur:id,nama_lengkap,gelar_depan,gelar_belakang']);

        // Filter by kategori
        if ($request->has('kategori_id')) {
            $query->where('kategori_id', $request->kategori_id);
        }

        // Filter by status
        if ($request->has('status')) {
            $query->where('status', $request->status);
        }

        // Filter by level
        if ($request->has('level')) {
            $query->where('level', $request->level);
        }

        // Filter by tipe
        if ($request->has('tipe')) {
            $query->where('tipe', $request->tipe);
        }

        // Filter by search term (judul atau deskripsi)
        if ($request->has('search')) {
            $query->where(function($q) use ($request) {
                $q->where('judul', 'like', '%' . $request->search . '%')
                  ->orWhere('deskripsi', 'like', '%' . $request->search . '%');
            });
        }

        // Filter by instructor (admin only)
        if (Auth::guard('sanctum')->user()->tokenCan('super_admin') && $request->has('instruktur_id')) {
            $query->where('admin_instruktur_id', $request->instruktur_id);
        }
```

```
    // Filter kursus by instruktur (if not super admin)
    if (Auth::guard('sanctum')->user()->tokenCan('instruktur')) {
        $query->where('admin_instruktur_id', Auth::id());
    }

    // Default ordering
    $orderBy = $request->order_by ?? 'created_at';
    $orderDir = $request->order_dir ?? 'desc';

    $kursus = $query->orderBy($orderBy, $orderDir)->paginate($request->per_page ?? 15);

    return response()->json([
        'status' => 'success',
        'data' => $kursus
    ]);
}

public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'kategori_id' => 'required|exists:kategori_kursus,id',
        'judul' => 'required|string|max:255',
        'deskripsi' => 'required|string',
        'tujuan_pembelajaran' => 'nullable|string',
        'sasaran_peserta' => 'nullable|string',
        'durasi_jam' => 'nullable|integer|min:1',
        'tanggal_buka_pendaftaran' => 'nullable|date',
        'tanggal_tutup_pendaftaran' => 'nullable|date|after_or_equal:tanggal_buka_pendaftaran',
        'tanggal_mulai_kursus' => 'nullable|date|after_or_equal:tanggal_tutup_pendaftaran',
        'tanggal_selesai_kursus' => 'nullable|date|after_or_equal:tanggal_mulai_kursus',
        'kuota_peserta' => 'nullable|integer|min:1',
        'level' => 'required|in:dasar,menengah,lanjut',
        'tipe' => 'required|in:daring,luring,hybrid',
        'status' => 'nullable|in:draft,aktif,nonaktif,selesai',
        'passing_grade' => 'nullable|numeric|min:0|max:100',
        'thumbnail' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
    ]);

    if ($validator->fails()) {
```

```
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
                'errors' => $validator->errors()
            ], 422);
        }

        $data = $request->all();
        $data['admin_instruktur_id'] = Auth::id();
        $data['kode_kursus'] = 'K' . date('Ym') . Str::random(5);
        $data['status'] = $request->status ?? 'draft';

        // Upload thumbnail
        if ($request->hasFile('thumbnail')) {
            $file = $request->file('thumbnail');
            $filename = time() . '_' . $file->getClientOriginalName();
            $path = $file->storeAs('kursus/thumbnail', $filename, 'public');
            $data['thumbnail'] = $filename;
        }

        $kursus = Kursus::create($data);

        return response()->json([
            'status' => 'success',
            'message' => 'Kursus berhasil dibuat',
            'data' => $kursus
        ], 201);
    }

    public function show($id)
    {
        $kursus = Kursus::with([
            'kategori:id,nama_kategori',
            'instruktur:id,nama_lengkap,gelar_depan,gelar_belakang',
            'modul:id,kursus_id,nama_modul,urutan',
            'prasyarat:id,kursus_id,kursus_prasyarat_id,deskripsi,is_wajib',
            'prasyarat.kursusPrasyarat:id,judul'
        ])->find($id);
```

```
        if (!$kursus) {
            return response()->json([
                'status' => 'error',
                'message' => 'Kursus tidak ditemukan'
            ], 404);
        }

        // Check authorization (only super admin or course instructor can see draft courses)
        $user = Auth::guard('sanctum')->user();
        if ($kursus->status === 'draft' && !($user->tokenCan('super_admin') || $kursus->admin_instruktur_id
=== $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        return response()->json([
            'status' => 'success',
            'data' => $kursus
        ]);
    }

    public function update(Request $request, $id)
    {
        $kursus = Kursus::find($id);

        if (!$kursus) {
            return response()->json([
                'status' => 'error',
                'message' => 'Kursus tidak ditemukan'
            ], 404);
        }

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
```

```
    ], 403);
}

$validator = Validator::make($request->all(), [
    'kategori_id' => 'sometimes|required|exists:kategori_kursus,id',
    'judul' => 'sometimes|required|string|max:255',
    'deskripsi' => 'sometimes|required|string',
    'tujuan_pembelajaran' => 'nullable|string',
    'sasaran_peserta' => 'nullable|string',
    'durasi_jam' => 'nullable|integer|min:1',
    'tanggal_buka_pendaftaran' => 'nullable|date',
    'tanggal_tutup_pendaftaran' => 'nullable|date|after_or_equal:tanggal_buka_pendaftaran',
    'tanggal_mulai_kursus' => 'nullable|date|after_or_equal:tanggal_tutup_pendaftaran',
    'tanggal_selesai_kursus' => 'nullable|date|after_or_equal:tanggal_mulai_kursus',
    'kuota_peserta' => 'nullable|integer|min:1',
    'level' => 'sometimes|required|in:dasar,menengah,lanjut',
    'tipe' => 'sometimes|required|in:daring,luring,hybrid',
    'status' => 'nullable|in:draft,aktif,nonaktif,selesai',
    'passing_grade' => 'nullable|numeric|min:0|max:100',
    'thumbnail' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
]);

if ($validator->fails()) {
    return response()->json([
        'status' => 'error',
        'message' => 'Validasi gagal',
        'errors' => $validator->errors()
    ], 422);
}

$data = $request->all();

// Upload thumbnail
if ($request->hasFile('thumbnail')) {
    // Delete old thumbnail
    if ($kursus->thumbnail) {
        Storage::disk('public')->delete('kursus/thumbnail/' . $kursus->thumbnail);
    }
```

```
        $file = $request->file('thumbnail');
        $filename = time() . '_' . $file->getClientOriginalName();
        $path = $file->storeAs('kursus/thumbnail', $filename, 'public');
        $data['thumbnail'] = $filename;
    }

    $kursus->update($data);

    return response()->json([
        'status' => 'success',
        'message' => 'Kursus berhasil diperbarui',
        'data' => $kursus
    ]);
}

public function destroy($id)
{
    $kursus = Kursus::find($id);

    if (!$kursus) {
        return response()->json([
            'status' => 'error',
            'message' => 'Kursus tidak ditemukan'
        ], 404);
    }

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Check if kursus has pendaftaran
    $pendaftaranCount = $kursus->pendaftaran()->count();
    if ($pendaftaranCount > 0) {
        return response()->json([
```

```
            'status' => 'error',
            'message' => 'Kursus tidak dapat dihapus karena memiliki ' . $pendaftaranCount . ' pendaftaran'
        ], 400);
    }

    // Delete thumbnail
    if ($kursus->thumbnail) {
        Storage::disk('public')->delete('kursus/thumbnail/' . $kursus->thumbnail);
    }

    $kursus->delete();

    return response()->json([
        'status' => 'success',
        'message' => 'Kursus berhasil dihapus'
    ]);
  }
}
```

Routes

php

```
// Modules/Kursus/Routes/api.php
use Illuminate\Support\Facades\Route;
use Modules\Kursus\Http\Controllers\KursusController;
use Modules\Kursus\Http\Controllers\PendaftaranController;

Route::middleware('auth:sanctum')->group(function () {
  // Kursus Routes
  Route::get('kursus', [KursusController::class, 'index']);
  Route::post('kursus', [KursusController::class, 'store'])
    ->middleware('ability:super_admin,instruktur');
  Route::get('kursus/{id}', [KursusController::class, 'show']);
  Route::put('kursus/{id}', [KursusController::class, 'update'])
    ->middleware('ability:super_admin,instruktur');
  Route::delete('kursus/{id}', [KursusController::class, 'destroy'])
    ->middleware('ability:super_admin,instruktur');

  // Pendaftaran Routes
  Route::get('pendaftaran', [PendaftaranController::class, 'index']);
```

```
    Route::post('kursus/{id}/daftar', [PendaftaranController::class, 'store'])
        ->middleware('ability:peserta');
    Route::put('pendaftaran/{id}/status', [PendaftaranController::class, 'updateStatus'])
        ->middleware('ability:super_admin,instruktur');
    Route::get('pendaftaran/peserta', [PendaftaranController::class, 'getByPeserta'])
        ->middleware('ability:peserta');
    Route::get('pendaftaran/kursus/{kursusId}', [PendaftaranController::class, 'getByKursus'])
        ->middleware('ability:super_admin,instruktur');
});
```

7. Pengembangan Modul Materi

php

```
// Modules/Materi/Http/Controllers/ModulController.php
namespace Modules\Materi\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Materi\Entities\Modul;
use Modules\Kursus\Entities\Kursus;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;

class ModulController extends Controller
{
    public function index($kursusId)
    {
        $kursus = Kursus::findOrFail($kursusId);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        $modul = Modul::where('kursus_id', $kursusId)
            ->with('materi')
```

```
            ->orderBy('urutan')
            ->get();

        return response()->json([
            'status' => 'success',
            'data' => [
                'kursus' => [
                    'id' => $kursus->id,
                    'judul' => $kursus->judul
                ],
                'modul' => $modul
            ]
        ]);
    }

    public function store(Request $request, $kursusId)
    {
        $kursus = Kursus::findOrFail($kursusId);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        $validator = Validator::make($request->all(), [
            'nama_modul' => 'required|string|max:255',
            'deskripsi' => 'nullable|string',
            'urutan' => 'nullable|integer|min:1',
            'is_published' => 'nullable|boolean'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
```

```
            'errors' => $validator->errors()
        ], 422);
    }

    // If urutan not provided, get last urutan + 1
    if (!$request->has('urutan')) {
        $lastUrutan = Modul::where('kursus_id', $kursusId)->max('urutan') ?? 0;
        $urutan = $lastUrutan + 1;
    } else {
        $urutan = $request->urutan;

        // Reorder existing modules if necessary
        $existingModul = Modul::where('kursus_id', $kursusId)
            ->where('urutan', '>=', $urutan)
            ->orderBy('urutan')
            ->get();

        foreach ($existingModul as $modul) {
            $modul->urutan += 1;
            $modul->save();
        }
    }

    $modul = new Modul([
        'nama_modul' => $request->nama_modul,
        'deskripsi' => $request->deskripsi,
        'urutan' => $urutan,
        'is_published' => $request->is_published ?? false
    ]);

    $modul->kursus()->associate($kursus);
    $modul->save();

    return response()->json([
        'status' => 'success',
        'message' => 'Modul berhasil dibuat',
        'data' => $modul
    ], 201);
}
```

```
public function show($id)
{
    $modul = Modul::with(['kursus:id,judul,admin_instruktur_id', 'materi' => function($query) {
        $query->orderBy('urutan');
    }])->findOrFail($id);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    return response()->json([
        'status' => 'success',
        'data' => $modul
    ]);
}

public function update(Request $request, $id)
{
    $modul = Modul::with('kursus:id,admin_instruktur_id')->findOrFail($id);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
        'nama_modul' => 'sometimes|required|string|max:255',
        'deskripsi' => 'nullable|string',
        'urutan' => 'nullable|integer|min:1',
```

```
        'is_published' => 'nullable|boolean'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    // Handle reordering if urutan changed
    if ($request->has('urutan') && $request->urutan != $modul->urutan) {
        $oldUrutan = $modul->urutan;
        $newUrutan = $request->urutan;

        if ($newUrutan > $oldUrutan) {
            // Moving down, shift up modules in between
            Modul::where('kursus_id', $modul->kursus_id)
                ->where('urutan', '>', $oldUrutan)
                ->where('urutan', '<=', $newUrutan)
                ->decrement('urutan');
        } else {
            // Moving up, shift down modules in between
            Modul::where('kursus_id', $modul->kursus_id)
                ->where('urutan', '>=', $newUrutan)
                ->where('urutan', '<', $oldUrutan)
                ->increment('urutan');
        }
    }

    $modul->fill($request->all());
    $modul->save();

    return response()->json([
        'status' => 'success',
        'message' => 'Modul berhasil diperbarui',
        'data' => $modul
    ]);
```

```
}

public function destroy($id)
{
    $modul = Modul::with(['kursus:id,admin_instruktur_id', 'materi'])->findOrFail($id);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Check if modul has quizzes
    $quizCount = $modul->quiz()->count();
    if ($quizCount > 0) {
        return response()->json([
            'status' => 'error',
            'message' => 'Modul tidak dapat dihapus karena memiliki ' . $quizCount . ' quiz'
        ], 400);
    }

    // Check if modul has materi
    $materiCount = $modul->materi()->count();
    if ($materiCount > 0) {
        return response()->json([
            'status' => 'error',
            'message' => 'Modul tidak dapat dihapus karena memiliki ' . $materiCount . ' materi'
        ], 400);
    }

    // Reorder remaining modules
    Modul::where('kursus_id', $modul->kursus_id)
        ->where('urutan', '>', $modul->urutan)
        ->decrement('urutan');

    $modul->delete();
```

```
        return response()->json([
            'status' => 'success',
            'message' => 'Modul berhasil dihapus'
        ]);
    }
}
```

php

```
// Modules/Materi/Http/Controllers/MateriController.php
namespace Modules\Materi\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Materi\Entities\Materi;
use Modules\Materi\Entities\Modul;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;

class MateriController extends Controller
{
    public function index($modulId)
    {
        $modul = Modul::with('kursus:id,judul,admin_instruktur_id')->findOrFail($modulId);

        $user = Auth::guard('sanctum')->user();

        // If peserta, check if they're enrolled in the course
        if ($user->tokenCan('peserta')) {
            $isEnrolled = $user->pendaftaranKursus()
                ->where('kursus_id', $modul->kursus_id)
                ->whereIn('status', ['disetujui', 'aktif'])
                ->exists();

            if (!$isEnrolled) {
                return response()->json([
                    'status' => 'error',
                    'message' => 'Anda belum terdaftar dalam kursus ini'
                ], 403);
```

```
        }

        // If module is not published, deny access
        if (!$modul->is_published) {
            return response()->json([
                'status' => 'error',
                'message' => 'Modul belum dipublikasikan'
            ], 403);
        }
    }
    // If admin/instruktur, check if they have access to the course
    else if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $materi = Materi::where('modul_id', $modulId)
        ->orderBy('urutan')
        ->get();

    if ($user->tokenCan('peserta')) {
        // Get progress for each materi
        $materi->each(function($item) use ($
```

Coba lagi
R

Lanjutkan

php

```
// Modules/Materi/Http/Controllers/MateriController.php (continued)
        $materi->each(function($item) use ($user) {
            $progress = $user->progresMateri()
                ->where('materi_id', $item->id)
                ->first();

            $item->progress = $progress ? [
                'is_selesai' => $progress->is_selesai,
                'progress_persen' => $progress->progress_persen,
```

```
                'tanggal_mulai' => $progress->tanggal_mulai,
                'tanggal_selesai' => $progress->tanggal_selesai,
                'durasi_belajar_menit' => $progress->durasi_belajar_menit
            ] : null;
        });
    }

    return response()->json([
        'status' => 'success',
        'data' => [
            'modul' => [
                'id' => $modul->id,
                'nama_modul' => $modul->nama_modul,
                'deskripsi' => $modul->deskripsi,
                'is_published' => $modul->is_published,
                'kursus' => [
                    'id' => $modul->kursus->id,
                    'judul' => $modul->kursus->judul
                ]
            ],
            'materi' => $materi
        ]
    ]);
}

public function store(Request $request, $modulId)
{
    $modul = Modul::with('kursus:id,admin_instruktur_id')->findOrFail($modulId);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
```

```
    'judul_materi' => 'required|string|max:255',
    'tipe_konten' => 'required|in:pdf,doc,video,audio,gambar,link,scorm',
    'file' => 'required_unless:tipe_konten,link|file|max:102400',
    'link' => 'required_if:tipe_konten,link|url',
    'deskripsi' => 'nullable|string',
    'durasi_menit' => 'nullable|integer|min:1',
    'is_wajib' => 'nullable|boolean',
    'urutan' => 'nullable|integer|min:1'
]);

if ($validator->fails()) {
    return response()->json([
        'status' => 'error',
        'message' => 'Validasi gagal',
        'errors' => $validator->errors()
    ], 422);
}

// If urutan not provided, get last urutan + 1
if (!$request->has('urutan')) {
    $lastUrutan = Materi::where('modul_id', $modulId)->max('urutan') ?? 0;
    $urutan = $lastUrutan + 1;
} else {
    $urutan = $request->urutan;

    // Reorder existing materi if necessary
    $existingMateri = Materi::where('modul_id', $modulId)
        ->where('urutan', '>=', $urutan)
        ->orderBy('urutan')
        ->get();

    foreach ($existingMateri as $materi) {
        $materi->urutan += 1;
        $materi->save();
    }
}

$materi = new Materi([
    'judul_materi' => $request->judul_materi,
```

```
            'tipe_konten' => $request->tipe_konten,
            'deskripsi' => $request->deskripsi,
            'durasi_menit' => $request->durasi_menit ?? 0,
            'is_wajib' => $request->is_wajib ?? true,
            'urutan' => $urutan,
            'published_at' => $request->has('published') && $request->published ? now() : null
        ]);

        $materi->modul()->associate($modul);

        // Handle file upload
        if ($request->tipe_konten !== 'link' && $request->hasFile('file')) {
            $file = $request->file('file');
            $filename = time() . '_' . $file->getClientOriginalName();

            // Store file based on content type
            $path = 'materi/' . $request->tipe_konten . '/' . $filename;
            Storage::disk('public')->put($path, file_get_contents($file));

            $materi->file_path = $path;
            $materi->ukuran_file = $file->getSize();
        } else if ($request->tipe_konten === 'link') {
            $materi->file_path = $request->link;
        }

        $materi->save();

        return response()->json([
            'status' => 'success',
            'message' => 'Materi berhasil ditambahkan',
            'data' => $materi
        ], 201);
    }

    public function show($id)
    {
        $materi = Materi::with('modul.kursus:id,judul,admin_instruktur_id')->findOrFail($id);
        $modul = $materi->modul;
```

```php
$user = Auth::guard('sanctum')->user();

// If peserta, check if they're enrolled in the course
if ($user->tokenCan('peserta')) {
    $isEnrolled = $user->pendaftaranKursus()
        ->where('kursus_id', $modul->kursus_id)
        ->whereIn('status', ['disetujui', 'aktif'])
        ->exists();

    if (!$isEnrolled) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda belum terdaftar dalam kursus ini'
        ], 403);
    }

    // If module or materi is not published, deny access
    if (!$modul->is_published || !$materi->published_at) {
        return response()->json([
            'status' => 'error',
            'message' => 'Materi belum dipublikasikan'
        ], 403);
    }

    // Record that student has started this materi
    $progresMateri = $user->progresMateri()
        ->where('materi_id', $materi->id)
        ->first();

    if (!$progresMateri) {
        $user->progresMateri()->create([
            'materi_id' => $materi->id,
            'is_selesai' => false,
            'progress_persen' => 0,
            'tanggal_mulai' => now(),
            'durasi_belajar_menit' => 0
        ]);
    }
```

```
        // Get progress data
        $progress = $user->progresMateri()
            ->where('materi_id', $materi->id)
            ->first();

        $materi->progress = $progress ? [
            'is_selesai' => $progress->is_selesai,
            'progress_persen' => $progress->progress_persen,
            'tanggal_mulai' => $progress->tanggal_mulai,
            'tanggal_selesai' => $progress->tanggal_selesai,
            'durasi_belajar_menit' => $progress->durasi_belajar_menit
        ] : null;
    }
    // If admin/instruktur, check if they have access to the course
    else if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Generate file URL if applicable
    if ($materi->tipe_konten !== 'link' && $materi->file_path) {
        $materi->file_url = url('storage/' . $materi->file_path);
    }

    return response()->json([
        'status' => 'success',
        'data' => $materi
    ]);
}

public function update(Request $request, $id)
{
    $materi = Materi::with('modul.kursus:id,admin_instruktur_id')->findOrFail($id);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $materi->modul->kursus->admin_instruktur_id === $user->id)) {
```

```
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
        'judul_materi' => 'sometimes|required|string|max:255',
        'deskripsi' => 'nullable|string',
        'durasi_menit' => 'nullable|integer|min:1',
        'is_wajib' => 'nullable|boolean',
        'urutan' => 'nullable|integer|min:1',
        'published' => 'nullable|boolean',
        'file' => 'nullable|file|max:102400',
        'link' => 'nullable|url'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    // Handle reordering if urutan changed
    if ($request->has('urutan') && $request->urutan != $materi->urutan) {
        $oldUrutan = $materi->urutan;
        $newUrutan = $request->urutan;

        if ($newUrutan > $oldUrutan) {
            // Moving down, shift up materi in between
            Materi::where('modul_id', $materi->modul_id)
                ->where('urutan', '>', $oldUrutan)
                ->where('urutan', '<=', $newUrutan)
                ->decrement('urutan');
        } else {
            // Moving up, shift down materi in between
            Materi::where('modul_id', $materi->modul_id)
```

```
            ->where('urutan', '>=', $newUrutan)
            ->where('urutan', '<', $oldUrutan)
            ->increment('urutan');
    }
}

// Handle file replacement if provided
if ($request->hasFile('file')) {
    // Delete old file
    if ($materi->file_path && $materi->tipe_konten !== 'link') {
        Storage::disk('public')->delete($materi->file_path);
    }

    $file = $request->file('file');
    $filename = time() . '_' . $file->getClientOriginalName();

    // Store file based on content type
    $path = 'materi/' . $materi->tipe_konten . '/' . $filename;
    Storage::disk('public')->put($path, file_get_contents($file));

    $materi->file_path = $path;
    $materi->ukuran_file = $file->getSize();
} else if ($materi->tipe_konten === 'link' && $request->has('link')) {
    $materi->file_path = $request->link;
}

// Update publication status
if ($request->has('published')) {
    if ($request->published && !$materi->published_at) {
        $materi->published_at = now();
    } else if (!$request->published && $materi->published_at) {
        $materi->published_at = null;
    }
}

$materi->fill($request->except(['file', 'link', 'published']));
$materi->save();

return response()->json([
```

```
            'status' => 'success',
            'message' => 'Materi berhasil diperbarui',
            'data' => $materi
        ]);
    }

    public function destroy($id)
    {
        $materi = Materi::with('modul.kursus:id,admin_instruktur_id')->findOrFail($id);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $materi->modul->kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        // Delete file if exists
        if ($materi->file_path && $materi->tipe_konten !== 'link') {
            Storage::disk('public')->delete($materi->file_path);
        }

        // Reorder remaining materi
        Materi::where('modul_id', $materi->modul_id)
            ->where('urutan', '>', $materi->urutan)
            ->decrement('urutan');

        $materi->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Materi berhasil dihapus'
        ]);
    }

    // Mark materi as completed
    public function markComplete(Request $request, $id)
```

```
{
    $materi = Materi::findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    if (!$user->tokenCan('peserta')) {
        return response()->json([
            'status' => 'error',
            'message' => 'Hanya peserta yang dapat menandai materi sebagai selesai'
        ], 403);
    }

    // Check if user is enrolled in the course
    $isEnrolled = $user->pendaftaranKursus()
        ->where('kursus_id', $materi->modul->kursus_id)
        ->whereIn('status', ['disetujui', 'aktif'])
        ->exists();

    if (!$isEnrolled) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda belum terdaftar dalam kursus ini'
        ], 403);
    }

    // Update or create progress
    $progress = $user->progresMateri()
        ->where('materi_id', $materi->id)
        ->first();

    if ($progress) {
        $progress->update([
            'is_selesai' => true,
            'progress_persen' => 100,
            'tanggal_selesai' => now(),
            'durasi_belajar_menit' => $request->durasi_belajar_menit ?? $progress->durasi_belajar_menit
        ]);
    } else {
        $progress = $user->progresMateri()->create([
```

```
            'materi_id' => $materi->id,
            'is_selesai' => true,
            'progress_persen' => 100,
            'tanggal_mulai' => now(),
            'tanggal_selesai' => now(),
            'durasi_belajar_menit' => $request->durasi_belajar_menit ?? 0
        ]);
    }

    return response()->json([
        'status' => 'success',
        'message' => 'Materi berhasil ditandai sebagai selesai',
        'data' => $progress
    ]);
}

// Update progress without completing
public function updateProgress(Request $request, $id)
{
    $materi = Materi::findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    if (!$user->tokenCan('peserta')) {
        return response()->json([
            'status' => 'error',
            'message' => 'Hanya peserta yang dapat memperbarui progres materi'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
        'progress_persen' => 'required|integer|min:0|max:100',
        'durasi_belajar_menit' => 'nullable|integer|min:0'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
```

```
            'errors' => $validator->errors()
        ], 422);
    }

    // Check if user is enrolled in the course
    $isEnrolled = $user->pendaftaranKursus()
        ->where('kursus_id', $materi->modul->kursus_id)
        ->whereIn('status', ['disetujui', 'aktif'])
        ->exists();

    if (!$isEnrolled) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda belum terdaftar dalam kursus ini'
        ], 403);
    }

    // Update or create progress
    $progress = $user->progresMateri()
        ->where('materi_id', $materi->id)
        ->first();

    if ($progress) {
        $progress->update([
            'progress_persen' => $request->progress_persen,
            'is_selesai' => $request->progress_persen == 100,
            'tanggal_selesai' => $request->progress_persen == 100 ? now() : null,
            'durasi_belajar_menit' => $request->durasi_belajar_menit ?? $progress->durasi_belajar_menit
        ]);
    } else {
        $progress = $user->progresMateri()->create([
            'materi_id' => $materi->id,
            'is_selesai' => $request->progress_persen == 100,
            'progress_persen' => $request->progress_persen,
            'tanggal_mulai' => now(),
            'tanggal_selesai' => $request->progress_persen == 100 ? now() : null,
            'durasi_belajar_menit' => $request->durasi_belajar_menit ?? 0
        ]);
    }
```

```
        return response()->json([
            'status' => 'success',
            'message' => 'Progres materi berhasil diperbarui',
            'data' => $progress
        ]);
    }
}
```

8. Pengembangan Modul Evaluasi

php

```
// Modules/Evaluasi/Http/Controllers/QuizController.php
namespace Modules\Evaluasi\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Evaluasi\Entities\Quiz;
use Modules\Evaluasi\Entities\SoalQuiz;
use Modules\Evaluasi\Entities\QuizResult;
use Modules\Materi\Entities\Modul;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;

class QuizController extends Controller
{
    public function index($modulId)
    {
        $modul = Modul::with('kursus:id,judul,admin_instruktur_id')->findOrFail($modulId);

        // Check authorization
        $user = Auth::guard('sanctum')->user();

        // If peserta, check if they're enrolled in the course
        if ($user->tokenCan('peserta')) {
            $isEnrolled = $user->pendaftaranKursus()
                ->where('kursus_id', $modul->kursus_id)
                ->whereIn('status', ['disetujui', 'aktif'])
                ->exists();
```

```
    if (!$isEnrolled) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda belum terdaftar dalam kursus ini'
        ], 403);
    }

    // Peserta only sees published quizzes
    $quizzes = Quiz::where('modul_id', $modulId)
        ->where('is_published', true)
        ->get(['id', 'judul_quiz', 'deskripsi', 'durasi_menit', 'passing_grade', 'max_attempt']);

    // Get user's quiz results for each quiz
    foreach ($quizzes as $quiz) {
        $results = QuizResult::where('quiz_id', $quiz->id)
            ->where('peserta_id', $user->id)
            ->orderBy('created_at', 'desc')
            ->get();

        $quiz->results = $results;
        $quiz->attempts = $results->count();
        $quiz->best_score = $results->max('nilai') ?? 0;
        $quiz->passed = $results->where('is_passed', true)->count() > 0;
    }
}
// If admin/instruktur, check if they have access to the course
else if ($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id) {
    $quizzes = Quiz::where('modul_id', $modulId)
        ->withCount('soal')
        ->get();
} else {
    return response()->json([
        'status' => 'error',
        'message' => 'Anda tidak memiliki akses'
    ], 403);
}

return response()->json([
    'status' => 'success',
```

```
            'data' => [
                'modul' => [
                    'id' => $modul->id,
                    'nama_modul' => $modul->nama_modul,
                    'kursus' => [
                        'id' => $modul->kursus->id,
                        'judul' => $modul->kursus->judul
                    ]
                ],
                'quizzes' => $quizzes
            ]
        ]);
    }

    public function store(Request $request, $modulId)
    {
        $modul = Modul::with('kursus:id,admin_instruktur_id')->findOrFail($modulId);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $modul->kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        $validator = Validator::make($request->all(), [
            'judul_quiz' => 'required|string|max:255',
            'deskripsi' => 'nullable|string',
            'durasi_menit' => 'required|integer|min:1',
            'bobot_nilai' => 'required|numeric|min:0|max:100',
            'passing_grade' => 'required|integer|min:0|max:100',
            'jumlah_soal' => 'nullable|integer|min:0',
            'random_soal' => 'nullable|boolean',
            'tampilkan_hasil' => 'nullable|boolean',
            'max_attempt' => 'required|integer|min:1',
            'is_published' => 'nullable|boolean'
        ]);
```

```
    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    $quiz = new Quiz($request->all());
    $quiz->modul_id = $modulId;
    $quiz->save();

    return response()->json([
        'status' => 'success',
        'message' => 'Quiz berhasil ditambahkan',
        'data' => $quiz
    ], 201);
}

public function show($id)
{
    $quiz = Quiz::with('modul.kursus:id,judul,admin_instruktur_id')->findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    // If peserta, check if they're enrolled in the course
    if ($user->tokenCan('peserta')) {
        $isEnrolled = $user->pendaftaranKursus()
            ->where('kursus_id', $quiz->modul->kursus_id)
            ->whereIn('status', ['disetujui', 'aktif'])
            ->exists();

        if (!$isEnrolled) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda belum terdaftar dalam kursus ini'
            ], 403);
        }
```

```
        // Peserta cannot access unpublished quiz
        if (!$quiz->is_published) {
            return response()->json([
                'status' => 'error',
                'message' => 'Quiz belum dipublikasikan'
            ], 403);
        }

        // Get user's quiz results
        $results = QuizResult::where('quiz_id', $quiz->id)
            ->where('peserta_id', $user->id)
            ->orderBy('created_at', 'desc')
            ->get();

        $quiz->results = $results;
        $quiz->attempts = $results->count();
        $quiz->best_score = $results->max('nilai') ?? 0;
        $quiz->passed = $results->where('is_passed', true)->count() > 0;

        // Check if user can take the quiz (max attempts not reached)
        $quiz->can_take = $quiz->attempts < $quiz->max_attempt || $quiz->max_attempt === 0;

        // Do not include soal for peserta
        unset($quiz->soal);
    }
    // If admin/instruktur, check if they have access to the course
    else if ($user->tokenCan('super_admin') || $quiz->modul->kursus->admin_instruktur_id === $user->id) {
        $quiz->load('soal');
    } else {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    return response()->json([
        'status' => 'success',
        'data' => $quiz
```

```
        ]);
    }

    public function update(Request $request, $id)
    {
        $quiz = Quiz::with('modul.kursus:id,admin_instruktur_id')->findOrFail($id);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $quiz->modul->kursus->admin_instruktur_id === $user->id))
{
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        $validator = Validator::make($request->all(), [
            'judul_quiz' => 'sometimes|required|string|max:255',
            'deskripsi' => 'nullable|string',
            'durasi_menit' => 'sometimes|required|integer|min:1',
            'bobot_nilai' => 'sometimes|required|numeric|min:0|max:100',
            'passing_grade' => 'sometimes|required|integer|min:0|max:100',
            'jumlah_soal' => 'nullable|integer|min:0',
            'random_soal' => 'nullable|boolean',
            'tampilkan_hasil' => 'nullable|boolean',
            'max_attempt' => 'sometimes|required|integer|min:1',
            'is_published' => 'nullable|boolean'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validasi gagal',
                'errors' => $validator->errors()
            ], 422);
        }

        $quiz->update($request->all());
```

```
        return response()->json([
            'status' => 'success',
            'message' => 'Quiz berhasil diperbarui',
            'data' => $quiz
        ]);
    }

    public function destroy($id)
    {
        $quiz = Quiz::with('modul.kursus:id,admin_instruktur_id')->findOrFail($id);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $quiz->modul->kursus->admin_instruktur_id === $user->id))
{
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        // Check if quiz has results
        $resultCount = $quiz->hasil()->count();
        if ($resultCount > 0) {
            return response()->json([
                'status' => 'error',
                'message' => 'Quiz tidak dapat dihapus karena memiliki ' . $resultCount . ' hasil pengerjaan'
            ], 400);
        }

        // Delete all quiz questions first
        $quiz->soal()->delete();

        // Delete quiz
        $quiz->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Quiz berhasil dihapus'
        ]);
```

```
}

// Start quiz attempt
public function startQuiz($id)
{
    $quiz = Quiz::with('modul.kursus:id,admin_instruktur_id')->findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    if (!$user->tokenCan('peserta')) {
        return response()->json([
            'status' => 'error',
            'message' => 'Hanya peserta yang dapat mengerjakan quiz'
        ], 403);
    }

    // Check if user is enrolled in the course
    $isEnrolled = $user->pendaftaranKursus()
        ->where('kursus_id', $quiz->modul->kursus_id)
        ->whereIn('status', ['disetujui', 'aktif'])
        ->exists();

    if (!$isEnrolled) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda belum terdaftar dalam kursus ini'
        ], 403);
    }

    // Check if quiz is published
    if (!$quiz->is_published) {
        return response()->json([
            'status' => 'error',
            'message' => 'Quiz belum dipublikasikan'
        ], 403);
    }

    // Check if max attempts reached
    $attemptCount = QuizResult::where('quiz_id', $quiz->id)
```

```
        ->where('peserta_id', $user->id)
        ->count();

    if ($attemptCount >= $quiz->max_attempt && $quiz->max_attempt !== 0) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda telah mencapai batas maksimal percobaan untuk quiz ini'
        ], 400);
    }

    // Get questions for the quiz
    if ($quiz->random_soal && $quiz->jumlah_soal > 0) {
        // Get random questions
        $soal = SoalQuiz::where('quiz_id', $quiz->id)
            ->inRandomOrder()
            ->limit($quiz->jumlah_soal)
            ->get(['id', 'pertanyaan', 'pilihan_a', 'pilihan_b', 'pilihan_c', 'pilihan_d', 'poin', 'tingkat_kesulitan']);
    } else {
        // Get all questions
        $soal = SoalQuiz::where('quiz_id', $quiz->id)
            ->get(['id', 'pertanyaan', 'pilihan_a', 'pilihan_b', 'pilihan_c', 'pilihan_d', 'poin', 'tingkat_kesulitan']);
    }

    if ($soal->count() === 0) {
        return response()->json([
            'status' => 'error',
            'message' => 'Quiz belum memiliki soal'
        ], 400);
    }

    // Create new quiz result
    $result = new QuizResult([
        'quiz_id' => $quiz->id,
        'peserta_id' => $user->id,
        'waktu_mulai' => now(),
        'attempt' => $attemptCount + 1
    ]);

    $result->save();
```

```
        return response()->json([
            'status' => 'success',
            'message' => 'Quiz dimulai',
            'data' => [
                'quiz' => [
                    'id' => $quiz->id,
                    'judul_quiz' => $quiz->judul_quiz,
                    'durasi_menit' => $quiz->durasi_menit,
                    'tampilkan_hasil' => $quiz->tampilkan_hasil,
                    'attempt' => $attemptCount + 1,
                    'max_attempt' => $quiz->max_attempt,
                    'waktu_mulai' => $result->waktu_mulai,
                    'batas_waktu' => $result->waktu_mulai->addMinutes($quiz->durasi_menit)
                ],
                'result_id' => $result->id,
                'soal' => $soal
            ]
        ]);
    }

    // Submit quiz answers
    public function submitQuiz(Request $request, $resultId)
    {
        $result = QuizResult::with(['quiz.soal', 'quiz.modul.kursus'])->findOrFail($resultId);

        $user = Auth::guard('sanctum')->user();

        if (!$user->tokenCan('peserta')) {
            return response()->json([
                'status' => 'error',
                'message' => 'Hanya peserta yang dapat mengerjakan quiz'
            ], 403);
        }

        // Check if this result belongs to this user
        if ($result->peserta_id !== $user->id) {
            return response()->json([
                'status' => 'error',
```

```
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Check if quiz already submitted
    if ($result->waktu_selesai !== null) {
        return response()->json([
            'status' => 'error',
            'message' => 'Quiz sudah disubmit sebelumnya'
        ], 400);
    }

    $validator = Validator::make($request->all(), [
        'jawaban' => 'required|array',
        'jawaban.*' => 'required'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    $jawaban = $request->jawaban;
    $quiz = $result->quiz;
    $soalQuiz = $quiz->soal;

    // Calculate score
    $jumlahBenar = 0;
    $jumlahSalah = 0;
    $totalPoin = 0;

    foreach ($soalQuiz as $soal) {
        if (isset($jawaban[$soal->id]) && $jawaban[$soal->id] === $soal->jawaban_benar) {
            $jumlahBenar++;
            $totalPoin += $soal->poin;
        } else {
```

```
            $jumlahSalah++;
        }
    }

    // Calculate score based on total points
    $totalMaxPoin = $soalQuiz->sum('poin');
    $nilai = $totalMaxPoin > 0 ? ($totalPoin / $totalMaxPoin) * 100 : 0;

    // Update result
    $result->update([
        'nilai' => $nilai,
        'jumlah_benar' => $jumlahBenar,
        'jumlah_salah' => $jumlahSalah,
        'is_passed' => $nilai >= $quiz->passing_grade,
        'waktu_selesai' => now(),
        'durasi_pengerjaan_menit' => now()->diffInMinutes($result->waktu_mulai)
    ]);

    return response()->json([
        'status' => 'success',
        'message' => 'Quiz berhasil diselesaikan',
        'data' => [
            'result' => $result,
            'pembahasan' => $quiz->tampilkan_hasil ? $soalQuiz->pluck('pembahasan', 'id') : null
        ]
    ]);
  }
}
```

9. Pengembangan Modul Kehadiran

php

```
// Modules/Kehadiran/Http/Controllers/SesiKehadiranController.php
namespace Modules\Kehadiran\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Kehadiran\Entities\SesiKehadiran;
use Modules\Kursus\Entities\Kursus;
use Illuminate\Support\Facades\Validator;
```

```php
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Str;

class SesiKehadiranController extends Controller
{
    public function index($kursusId)
    {
        $kursus = Kursus::findOrFail($kursusId);

        // Check authorization
        $user = Auth::guard('sanctum')->user();

        // If peserta, check if they're enrolled in the course
        if ($user->tokenCan('peserta')) {
            $isEnrolled = $user->pendaftaranKursus()
                ->where('kursus_id', $kursusId)
                ->whereIn('status', ['disetujui', 'aktif'])
                ->exists();

            if (!$isEnrolled) {
                return response()->json([
                    'status' => 'error',
                    'message' => 'Anda belum terdaftar dalam kursus ini'
                ], 403);
            }

            $sesi = SesiKehadiran::where('kursus_id', $kursusId)
                ->whereIn('status', ['scheduled', 'ongoing'])
                ->orderBy('tanggal')
                ->orderBy('waktu_mulai')
                ->get();

            // Get kehadiran data for this peserta
            foreach ($sesi as $s) {
                $kehadiran = $user->kehadiran()
                    ->where('sesi_id', $s->id)
                    ->first();

                $s->kehadiran = $kehadiran;
```

```
        }
    }
    // If admin/instruktur, check if they have access to the course
    else if ($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id) {
        $sesi = SesiKehadiran::where('kursus_id', $kursusId)
            ->orderBy('tanggal')
            ->orderBy('waktu_mulai')
            ->get();

        // Get all kehadiran for each sesi
        foreach ($sesi as $s) {
            $kehadiran = $s->kehadiran()
                ->with('peserta:id,nama_lengkap,nip')
                ->get();

            $s->kehadiran = $kehadiran;
            $s->hadir_count = $kehadiran->whereIn('status', ['hadir', 'terlambat'])->count();
            $s->tidak_hadir_count = $kehadiran->where('status', 'tidak_hadir')->count();
            $s->izin_count = $kehadiran->whereIn('status', ['izin', 'sakit'])->count();
        }
    } else {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    return response()->json([
        'status' => 'success',
        'data' => [
            'kursus' => [
                'id' => $kursus->id,
                'judul' => $kursus->judul
            ],
            'sesi' => $sesi
        ]
    ]);
}
```

```php
public function store(Request $request, $kursusId)
{
    $kursus = Kursus::findOrFail($kursusId);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
        'pertemuan_ke' => 'required|integer|min:1',
        'tanggal' => 'required|date',
        'waktu_mulai' => 'required|date_format:H:i',
        'waktu_selesai' => 'required|date_format:H:i|after:waktu_mulai',
        'durasi_berlaku_menit' => 'nullable|integer|min:1',
        'status' => 'nullable|in:scheduled,ongoing,completed,cancelled'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    $sesi = new SesiKehadiran($request->all());
    $sesi->kursus_id = $kursusId;
    $sesi->status = $request->status ?? 'scheduled';
    $sesi->durasi_berlaku_menit = $request->durasi_berlaku_menit ?? 30;

    // Generate QR codes
    $sesi->qr_code_checkin = Str::random(32);
    $sesi->qr_code_checkout = Str::random(32);
```

```
        $sesi->save();

        return response()->json([
            'status' => 'success',
            'message' => 'Sesi kehadiran berhasil ditambahkan',
            'data' => $sesi
        ], 201);
    }

    public function show($id)
    {
        $sesi = SesiKehadiran::with(['kursus:id,judul,admin_instruktur_id'])->findOrFail($id);

        $user = Auth::guard('sanctum')->user();

        // If peserta, check if they're enrolled in the course
        if ($user->tokenCan('peserta')) {
            $isEnrolled = $user->pendaftaranKursus()
                ->where('kursus_id', $sesi->kursus_id)
                ->whereIn('status', ['disetujui', 'aktif'])
                ->exists();

            if (!$isEnrolled) {
                return response()->json([
                    'status' => 'error',
                    'message' => 'Anda belum terdaftar dalam kursus ini'
                ], 403);
            }

            // Get kehadiran data for this peserta
            $kehadiran = $user->kehadiran()
                ->where('sesi_id', $sesi->id)
                ->first();

            $sesi->kehadiran = $kehadiran;

            // Only return QR code if the session is ongoing and within allowed time
            if ($sesi->status === 'ongoing' && now()->between(
                    now()->parse($sesi->tanggal . ' ' . $sesi->waktu_mulai)->subMinutes(15),
```

```
                now()->parse($sesi->tanggal . ' ' . $sesi->waktu_selesai)->addMinutes(15)
            )) {
            $sesi->qr_code_checkin_url = route('kehadiran.scan-checkin', ['token' => $sesi->qr_code_checkin]);
            $sesi->qr_code_checkout_url = route('kehadiran.scan-checkout', ['token' => $sesi->qr_code_checkout]);
        } else {
            $sesi->makeHidden(['qr_code_checkin', 'qr_code_checkout']);
        }
    }
    // If admin/instruktur, check if they have access to the course
    else if ($user->tokenCan('super_admin') || $sesi->kursus->admin_instruktur_id === $user->id) {
        // Get all kehadiran for this sesi
        $kehadiran = $sesi->kehadiran()
            ->with('peserta:id,nama_lengkap,nip')
            ->get();

        $sesi->kehadiran = $kehadiran;
        $sesi->hadir_count = $kehadiran->whereIn('status', ['hadir', 'terlambat'])->count();
        $sesi->tidak_hadir_count = $kehadiran->where('status', 'tidak_hadir')->count();
        $sesi->izin_count = $kehadiran->whereIn('status', ['izin', 'sakit'])->count();

        // Generate QR code URLs
        $sesi->qr_code_checkin_url = route('kehadiran.scan-checkin', ['token' => $sesi->qr_code_checkin]);
        $sesi->qr_code_checkout_url = route('kehadiran.scan-checkout', ['token' => $sesi->qr_code_checkout]);
    } else {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    return response()->json([
        'status' => 'success',
        'data' => $sesi
    ]);
}

public function update(Request $request, $id)
```

```
{
    $sesi = SesiKehadiran::with(['kursus:id,admin_instruktur_id'])->findOrFail($id);

    // Check authorization
    $user = Auth::guard('sanctum')->user();
    if (!($user->tokenCan('super_admin') || $sesi->kursus->admin_instruktur_id === $user->id)) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    $validator = Validator::make($request->all(), [
        'pertemuan_ke' => 'sometimes|required|integer|min:1',
        'tanggal' => 'sometimes|required|date',
        'waktu_mulai' => 'sometimes|required|date_format:H:i',
        'waktu_selesai' => 'sometimes|required|date_format:H:i|after:waktu_mulai',
        'durasi_berlaku_menit' => 'nullable|integer|min:1',
        'status' => 'nullable|in:scheduled,ongoing,completed,cancelled',
        'regenerate_qr' => 'nullable|boolean'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'status' => 'error',
            'message' => 'Validasi gagal',
            'errors' => $validator->errors()
        ], 422);
    }

    // Regenerate QR codes if requested
    if ($request->regenerate_qr) {
        $sesi->qr_code_checkin = Str::random(32);
        $sesi->qr_code_checkout = Str::random(32);
    }

    $sesi->update($request->except('regenerate_qr'));

    return response()->json([
```

```
            'status' => 'success',
            'message' => 'Sesi kehadiran berhasil diperbarui',
            'data' => $sesi
        ]);
    }

    public function destroy($id)
    {
        $sesi = SesiKehadiran::with(['kursus:id,admin_instruktur_id'])->findOrFail($id);

        // Check authorization
        $user = Auth::guard('sanctum')->user();
        if (!($user->tokenCan('super_admin') || $sesi->kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        // Check if sesi has kehadiran records
        $kehadiranCount = $sesi->kehadiran()->count();
        if ($kehadiranCount > 0) {
            return response()->json([
                'status' => 'error',
                'message' => 'Sesi tidak dapat dihapus karena memiliki ' . $kehadiranCount . ' rekaman
kehadiran'
            ], 400);
        }

        $sesi->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Sesi kehadiran berhasil dihapus'
        ]);
    }
}
```

10. Pengembangan Modul Sertifikat

php

```php
// Modules/Sertifikat/Http/Controllers/SertifikatController.php
namespace Modules\Sertifikat\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Modules\Sertifikat\Entities\Sertifikat;
use Modules\Sertifikat\Entities\TemplateSertifikat;
use Modules\Kursus\Entities\Kursus;
use Modules\Kursus\Entities\PendaftaranKursus;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
use PDF;
use Carbon\Carbon;

class SertifikatController extends Controller
{
    public function index(Request $request)
    {
        $user = Auth::guard('sanctum')->user();

        // Filter by peserta (if peserta, only show their certificates)
        if ($user->tokenCan('peserta')) {
            $sertifikat = Sertifikat::with(['kursus:id,judul', 'template:id,nama_template'])
                ->where('peserta_id', $user->id)
                ->orderBy('created_at', 'desc')
                ->get();
        }
        // Filter by kursus (admin/instruktur)
        else if ($user->tokenCan('super_admin') || $user->tokenCan('instruktur')) {
            $query = Sertifikat::with(['peserta:id,nama_lengkap,nip', 'kursus:id,judul',
'template:id,nama_template']);

            // Filter by kursus
            if ($request->has('kursus_id')) {
                $query->where('kursus_id', $request->kursus_id);
```

```
            // If instruktur, check if they can access this course
            if ($user->tokenCan('instruktur')) {
                $kursus = Kursus::find($request->kursus_id);
                if (!$kursus || $kursus->admin_instruktur_id !== $user->id) {
                    return response()->json([
                        'status' => 'error',
                        'message' => 'Anda tidak memiliki akses'
                    ], 403);
                }
            }
        } else if ($user->tokenCan('instruktur')) {
            // Instruktur can only see certificates for courses they teach
            $kursusIds = Kursus::where('admin_instruktur_id', $user->id)->pluck('id');
            $query->whereIn('kursus_id', $kursusIds);
        }

        // Filter by peserta
        if ($request->has('peserta_id')) {
            $query->where('peserta_id', $request->peserta_id);
        }

        $sertifikat = $query->orderBy('created_at', 'desc')->get();
    } else {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    return response()->json([
        'status' => 'success',
        'data' => $sertifikat
    ]);
}

public function generate(Request $request, $kursusId)
{
    $kursus = Kursus::findOrFail($kursusId);
```

```php
// Check authorization
$user = Auth::guard('sanctum')->user();
if (!($user->tokenCan('super_admin') || $kursus->admin_instruktur_id === $user->id)) {
    return response()->json([
        'status' => 'error',
        'message' => 'Anda tidak memiliki akses'
    ], 403);
}

$validator = Validator::make($request->all(), [
    'template_id' => 'required|exists:template_sertifikat,id',
    'peserta_ids' => 'nullable|array',
    'peserta_ids.*' => 'exists:peserta,id',
    'nama_penandatangan' => 'required|string',
    'jabatan_penandatangan' => 'required|string'
]);

if ($validator->fails()) {
    return response()->json([
        'status' => 'error',
        'message' => 'Validasi gagal',
        'errors' => $validator->errors()
    ], 422);
}

// Get template
$template = TemplateSertifikat::findOrFail($request->template_id);

// Get eligible peserta (completed course with passing grade)
$query = PendaftaranKursus::where('kursus_id', $kursusId)
    ->where('status', 'selesai')
    ->whereNotNull('nilai_akhir')
    ->where('nilai_akhir', '>=', $kursus->passing_grade);

// Filter by peserta_ids if provided
if ($request->has('peserta_ids') && !empty($request->peserta_ids)) {
    $query->whereIn('peserta_id', $request->peserta_ids);
}
```

```
$pendaftaran = $query->with('peserta')->get();

if ($pendaftaran->isEmpty()) {
    return response()->json([
        'status' => 'error',
        'message' => 'Tidak ada peserta yang memenuhi syarat untuk mendapatkan sertifikat'
    ], 404);
}

$generated = 0;
$errors = [];
$sertifikat = [];

// Generate certificate for each eligible peserta
foreach ($pendaftaran as $daftar) {
    // Check if certificate already exists
    $existingSertifikat = Sertifikat::where('peserta_id', $daftar->peserta_id)
        ->where('kursus_id', $kursusId)
        ->first();

    if ($existingSertifikat) {
        $errors[] = "Sertifikat untuk peserta {$daftar->peserta->nama_lengkap} sudah ada";
        continue;
    }

    try {
        // Generate unique certificate number
        $nomor = 'SERT/' . $kursus->kode_kursus . '/' . date('Y') . '/' . Str::random(5);

        // Generate QR code for verification
        $qrCode = Str::random(32);

        // Create certificate record
        $cert = Sertifikat::create([
            'peserta_id' => $daftar->peserta_id,
            'kursus_id' => $kursusId,
            'template_id' => $template->id,
            'nomor_sertifikat' => $nomor,
            'tanggal_terbit' => Carbon::now(),
```

```
                'qr_code' => $qrCode,
                'nama_penandatangan' => $request->nama_penandatangan,
                'jabatan_penandatangan' => $request->jabatan_penandatangan,
                'is_sent_email' => false
            ]);

            // Generate PDF certificate (implementation depends on how you want to generate the certificate)
            // This is a placeholder - you would need to implement PDF generation based on the template
            $pdf = PDF::loadView('sertifikat::pdf_template', [
                'sertifikat' => $cert,
                'template' => $template,
                'kursus' => $kursus,
                'peserta' => $daftar->peserta
            ]);

            $filename = 'sertifikat_' . $daftar->peserta_id . '_' . $kursusId . '.pdf';
            $path = 'sertifikat/' . $filename;
            Storage::disk('public')->put($path, $pdf->output());

            $cert->file_path = $filename;
            $cert->save();

            $sertifikat[] = $cert;
            $generated++;
        } catch (\Exception $e) {
            $errors[] = "Error generating certificate for {$daftar->peserta->nama_lengkap}: " . $e->getMessage();
        }
    }

    return response()->json([
        'status' => 'success',
        'message' => "Berhasil membuat {$generated} sertifikat" . (count($errors) > 0 ? " dengan " . count($errors) . " error" : ""),
        'data' => [
            'sertifikat' => $sertifikat,
            'errors' => $errors
        ]
    ]);
}
```

```php
public function show($id)
{
    $sertifikat = Sertifikat::with([
        'peserta:id,nama_lengkap,nip',
        'kursus:id,judul,admin_instruktur_id',
        'template:id,nama_template'
    ])->findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    // Check authorization
    if ($user->tokenCan('peserta')) {
        // Peserta can only view their own certificates
        if ($sertifikat->peserta_id !== $user->id) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }
    } else if ($user->tokenCan('instruktur')) {
        // Instruktur can only view certificates for their courses
        if ($sertifikat->kursus->admin_instruktur_id !== $user->id) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }
    } else if (!$user->tokenCan('super_admin')) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Generate file URL
    if ($sertifikat->file_path) {
        $sertifikat->file_url = url('storage/sertifikat/' . $sertifikat->file_path);
    }
```

```
    // Generate verification URL
    $sertifikat->verification_url = route('sertifikat.verify', ['code' => $sertifikat->qr_code]);

    return response()->json([
        'status' => 'success',
        'data' => $sertifikat
    ]);
}

public function verify($code)
{
    $sertifikat = Sertifikat::with([
        'peserta:id,nama_lengkap,nip',
        'kursus:id,judul'
    ])->where('qr_code', $code)->first();

    if (!$sertifikat) {
        return response()->json([
            'status' => 'error',
            'message' => 'Sertifikat tidak valid atau tidak ditemukan'
        ], 404);
    }

    return response()->json([
        'status' => 'success',
        'message' => 'Sertifikat terverifikasi',
        'data' => [
            'nomor_sertifikat' => $sertifikat->nomor_sertifikat,
            'nama_peserta' => $sertifikat->peserta->nama_lengkap,
            'nip' => $sertifikat->peserta->nip,
            'nama_kursus' => $sertifikat->kursus->judul,
            'tanggal_terbit' => $sertifikat->tanggal_terbit->format('d F Y'),
            'nama_penandatangan' => $sertifikat->nama_penandatangan,
            'jabatan_penandatangan' => $sertifikat->jabatan_penandatangan
        ]
    ]);
}
```

```
    public function sendEmail($id)
    {
        $sertifikat = Sertifikat::with(['peserta', 'k
```

Coba lagi
R

Lanjutkan

php

```
// Modules/Sertifikat/Http/Controllers/SertifikatController.php (continued)
    public function sendEmail($id)
    {
        $sertifikat = Sertifikat::with(['peserta', 'kursus:id,judul,admin_instruktur_id'])->findOrFail($id);

        $user = Auth::guard('sanctum')->user();

        // Check authorization
        if (!($user->tokenCan('super_admin') || $sertifikat->kursus->admin_instruktur_id === $user->id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }

        // Check if peserta has email
        if (!$sertifikat->peserta->email) {
            return response()->json([
                'status' => 'error',
                'message' => 'Peserta tidak memiliki alamat email'
            ], 400);
        }

        // Check if file exists
        if (!$sertifikat->file_path || !Storage::disk('public')->exists('sertifikat/' . $sertifikat->file_path)) {
            return response()->json([
                'status' => 'error',
                'message' => 'File sertifikat tidak ditemukan'
            ], 404);
        }
```

```
    try {
        // Send email with certificate (implementation depends on your email setup)
        Mail::to($sertifikat->peserta->email)->send(new SertifikatEmail($sertifikat));

        $sertifikat->is_sent_email = true;
        $sertifikat->save();

        return response()->json([
            'status' => 'success',
            'message' => 'Sertifikat berhasil dikirim via email'
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'status' => 'error',
            'message' => 'Gagal mengirim email: ' . $e->getMessage()
        ], 500);
    }
}

public function download($id)
{
    $sertifikat = Sertifikat::findOrFail($id);

    $user = Auth::guard('sanctum')->user();

    // Check authorization
    if ($user->tokenCan('peserta')) {
        // Peserta can only download their own certificates
        if ($sertifikat->peserta_id !== $user->id) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }
    } else if ($user->tokenCan('instruktur')) {
        // Instruktur can only download certificates for their courses
        $kursus = Kursus::find($sertifikat->kursus_id);
        if (!$kursus || $kursus->admin_instruktur_id !== $user->id) {
            return response()->json([
```

```
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ], 403);
        }
    } else if (!$user->tokenCan('super_admin')) {
        return response()->json([
            'status' => 'error',
            'message' => 'Anda tidak memiliki akses'
        ], 403);
    }

    // Check if file exists
    if (!$sertifikat->file_path || !Storage::disk('public')->exists('sertifikat/' . $sertifikat->file_path)) {
        return response()->json([
            'status' => 'error',
            'message' => 'File sertifikat tidak ditemukan'
        ], 404);
    }

    // Return file for download
    return Storage::disk('public')->download('sertifikat/' . $sertifikat->file_path, 'sertifikat_' . $sertifikat->nomor_sertifikat . '.pdf');
  }
}
```

11. API Documentation

Untuk dokumentasi API, kita bisa menggunakan Swagger/OpenAPI dengan menggunakan paket Laravel Swagger. Berikut langkah-langkah untuk mengimplementasikannya:

Instalasi Laravel Swagger

bash

```
composer require "darkaonline/l5-swagger"
php artisan vendor:publish --provider "L5Swagger\L5SwaggerServiceProvider"
```

Contoh Dokumentasi API untuk Endpoint Login

php

```
/**
 * @OA\Info(
 *    title="API E-Learning BPSDM",
```

```
*    version="1.0.0",
*    description="API Documentation for BPSDM E-Learning System"
* )
*/

/**
* @OA\Post(
*    path="/api/auth/login/admin",
*    tags={"Authentication"},
*    summary="Login as admin or instructor",
*    description="Returns admin user data and access token",
*    operationId="loginAdmin",
*    @OA\RequestBody(
*       required=true,
*       @OA\JsonContent(
*          required={"username", "password"},
*          @OA\Property(property="username", type="string", example="admin"),
*          @OA\Property(property="password", type="string", format="password", example="password")
*       )
*    ),
*    @OA\Response(
*       response=200,
*       description="Successful login",
*       @OA\JsonContent(
*          @OA\Property(property="status", type="string", example="success"),
*          @OA\Property(property="message", type="string", example="Login berhasil"),
*          @OA\Property(
*             property="data",
*             type="object",
*             @OA\Property(
*                property="user",
*                type="object",
*                @OA\Property(property="id", type="integer", example=1),
*                @OA\Property(property="username", type="string", example="admin"),
*                @OA\Property(property="email", type="string", example="admin@example.com"),
*                @OA\Property(property="role", type="string", example="super_admin")
*             ),
*             @OA\Property(property="token", type="string", example="1|a1b2c3d4e5f6g7h8i9j0"),
*             @OA\Property(property="token_type", type="string", example="Bearer")
```

```
     *         )
     *     )
     *   ),
     *   @OA\Response(
     *     response=422,
     *     description="Validation error",
     *     @OA\JsonContent(
     *       @OA\Property(property="status", type="string", example="error"),
     *       @OA\Property(property="message", type="string", example="Kredensial yang diberikan tidak cocok dengan catatan kami.")
     *     )
     *   )
     * )
     */
```

Generate Swagger Documentation

bash

```
php artisan l5-swagger:generate
```

Kemudian akses dokumentasi API di URL: `http://your-app.test/api/documentation`

12. Testing

PHPUnit Test untuk API Autentikasi

php

```
// Modules/Auth/Tests/Feature/AuthApiTest.php
namespace Modules\Auth\Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Modules\Auth\Entities\AdminInstruktur;
use Modules\Auth\Entities\Peserta;
use Modules\Master\Entities\OPD;

class AuthApiTest extends TestCase
{
    use RefreshDatabase;

    public function testAdminLogin()
    {
```

```
    $admin = AdminInstruktur::factory()->create([
        'username' => 'admin_test',
        'email' => 'admin@test.com',
        'password' => bcrypt('password'),
        'role' => 'super_admin'
    ]);

    $response = $this->postJson('/api/auth/login/admin', [
        'username' => 'admin_test',
        'password' => 'password'
    ]);

    $response->assertStatus(200)
        ->assertJsonStructure([
            'status',
            'message',
            'data' => [
                'user',
                'token',
                'token_type'
            ]
        ])
        ->assertJson([
            'status' => 'success',
            'message' => 'Login berhasil'
        ]);

    $this->assertNotNull($response->json('data.token'));
}

public function testPesertaLogin()
{
    $opd = OPD::factory()->create();

    $peserta = Peserta::factory()->create([
        'opd_id' => $opd->id,
        'username' => 'peserta_test',
        'email' => 'peserta@test.com',
        'password' => bcrypt('password')
```

```
        ]);

        $response = $this->postJson('/api/auth/login/peserta', [
            'username' => 'peserta_test',
            'password' => 'password'
        ]);

        $response->assertStatus(200)
            ->assertJsonStructure([
                'status',
                'message',
                'data' => [
                    'user',
                    'token',
                    'token_type'
                ]
            ])
            ->assertJson([
                'status' => 'success',
                'message' => 'Login berhasil'
            ]);

        $this->assertNotNull($response->json('data.token'));
    }

    public function testFailedLogin()
    {
        $response = $this->postJson('/api/auth/login/admin', [
            'username' => 'nonexistent',
            'password' => 'wrong'
        ]);

        $response->assertStatus(422)
            ->assertJsonStructure([
                'message',
                'errors' => [
                    'username'
                ]
            ]);
```

```
    }

    public function testLogout()
    {
        $admin = AdminInstruktur::factory()->create([
            'password' => bcrypt('password')
        ]);

        $loginResponse = $this->postJson('/api/auth/login/admin', [
            'username' => $admin->username,
            'password' => 'password'
        ]);

        $token = $loginResponse->json('data.token');

        $response = $this->withHeader('Authorization', 'Bearer ' . $token)
            ->postJson('/api/auth/logout');

        $response->assertStatus(200)
            ->assertJson([
                'status' => 'success',
                'message' => 'Logout berhasil'
            ]);

        // Try to access protected route with the token
        $profileResponse = $this->withHeader('Authorization', 'Bearer ' . $token)
            ->getJson('/api/auth/profile');

        $profileResponse->assertStatus(401);
    }
}
```

Test untuk API Kursus

php

```
// Modules/Kursus/Tests/Feature/KursusApiTest.php
namespace Modules\Kursus\Tests\Feature;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
```

```php
use Modules\Auth\Entities\AdminInstruktur;
use Modules\Master\Entities\KategoriKursus;
use Modules\Kursus\Entities\Kursus;
use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;

class KursusApiTest extends TestCase
{
    use RefreshDatabase;

    protected $admin;
    protected $instructor;
    protected $kategori;

    protected function setUp(): void
    {
        parent::setUp();

        $this->admin = AdminInstruktur::factory()->create([
            'role' => 'super_admin',
            'password' => bcrypt('password')
        ]);

        $this->instructor = AdminInstruktur::factory()->create([
            'role' => 'instruktur',
            'password' => bcrypt('password')
        ]);

        $this->kategori = KategoriKursus::factory()->create();

        // Login as admin
        $loginResponse = $this->postJson('/api/auth/login/admin', [
            'username' => $this->admin->username,
            'password' => 'password'
        ]);

        $this->adminToken = $loginResponse->json('data.token');

        // Login as instructor
```

```
    $instructorLoginResponse = $this->postJson('/api/auth/login/admin', [
        'username' => $this->instructor->username,
        'password' => 'password'
    ]);

    $this->instructorToken = $instructorLoginResponse->json('data.token');
}

public function testAdminCanCreateKursus()
{
    Storage::fake('public');

    $thumbnail = UploadedFile::fake()->image('thumbnail.jpg');

    $response = $this->withHeader('Authorization', 'Bearer ' . $this->adminToken)
        ->postJson('/api/kursus', [
            'kategori_id' => $this->kategori->id,
            'judul' => 'Kursus Test',
            'deskripsi' => 'Deskripsi kursus test',
            'tujuan_pembelajaran' => 'Tujuan pembelajaran test',
            'level' => 'dasar',
            'tipe' => 'daring',
            'thumbnail' => $thumbnail,
            'durasi_jam' => 20,
            'tanggal_mulai_kursus' => now()->addDays(10)->format('Y-m-d'),
            'tanggal_selesai_kursus' => now()->addDays(30)->format('Y-m-d'),
            'kuota_peserta' => 30,
            'passing_grade' => 70
        ]);

    $response->assertStatus(201)
        ->assertJson([
            'status' => 'success',
            'message' => 'Kursus berhasil dibuat'
        ]);

    $this->assertDatabaseHas('kursus', [
        'judul' => 'Kursus Test',
        'admin_instruktur_id' => $this->admin->id,
```

```
            'status' => 'draft'
        ]);

        Storage::disk('public')->assertExists('kursus/thumbnail/' . $response->json('data.thumbnail'));
    }

    public function testInstructorCanOnlySeeOwnCourses()
    {
        // Create a course by admin
        $adminCourse = Kursus::factory()->create([
            'admin_instruktur_id' => $this->admin->id,
            'kategori_id' => $this->kategori->id
        ]);

        // Create a course by instructor
        $instructorCourse = Kursus::factory()->create([
            'admin_instruktur_id' => $this->instructor->id,
            'kategori_id' => $this->kategori->id
        ]);

        // Instructor requests all courses
        $response = $this->withHeader('Authorization', 'Bearer ' . $this->instructorToken)
            ->getJson('/api/kursus');

        $response->assertStatus(200);

        // Instructor should only see their own course
        $this->assertEquals(1, count($response->json('data.data')));
        $this->assertEquals($instructorCourse->id, $response->json('data.data.0.id'));

        // Admin requests all courses
        $adminResponse = $this->withHeader('Authorization', 'Bearer ' . $this->adminToken)
            ->getJson('/api/kursus');

        $adminResponse->assertStatus(200);

        // Admin should see all courses
        $this->assertEquals(2, count($adminResponse->json('data.data')));
    }
```

```php
    public function testInstructorCannotEditOtherInstructorsCourses()
    {
        // Create a course by another instructor
        $otherInstructor = AdminInstruktur::factory()->create(['role' => 'instruktur']);
        $otherCourse = Kursus::factory()->create([
            'admin_instruktur_id' => $otherInstructor->id,
            'kategori_id' => $this->kategori->id
        ]);

        // Try to edit other instructor's course
        $response = $this->withHeader('Authorization', 'Bearer ' . $this->instructorToken)
            ->putJson('/api/kursus/' . $otherCourse->id, [
                'judul' => 'Updated Judul'
            ]);

        $response->assertStatus(403)
            ->assertJson([
                'status' => 'error',
                'message' => 'Anda tidak memiliki akses'
            ]);

        // Admin can edit any course
        $adminResponse = $this->withHeader('Authorization', 'Bearer ' . $this->adminToken)
            ->putJson('/api/kursus/' . $otherCourse->id, [
                'judul' => 'Admin Updated Judul'
            ]);

        $adminResponse->assertStatus(200)
            ->assertJson([
                'status' => 'success',
                'message' => 'Kursus berhasil diperbarui',
                'data' => [
                    'judul' => 'Admin Updated Judul'
                ]
            ]);
    }
}
```

13. Deployment

Konfigurasi Server

1. **Server Requirements**:
   - PHP >= 8.0
   - MySQL >= 5.7 atau MariaDB >= 10.3
   - Composer
   - Web Server (Nginx atau Apache)
2. **Konfigurasi PHP**:

ini

```ini
upload_max_filesize = 20M
post_max_size = 20M
memory_limit = 256M
max_execution_time = 120
```

3. **Konfigurasi Nginx**:

nginx

```nginx
server {
    listen 80;
    server_name api.elearning.bpsdm.example.com;
    root /var/www/elearning-bpsdm-api/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;
```

```
    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php8.0-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}
```

Proses Deployment

1. **Clone Repository**:

bash
```
git clone https://github.com/your-repo/bpsdm-elearning-api.git
cd bpsdm-elearning-api
```

2. **Install Dependencies**:

bash
```
composer install --optimize-autoloader --no-dev
```

3. **Konfigurasi Environment**:

bash
```
cp .env.example .env
php artisan key:generate
```

4. **Edit file .env**:
```
APP_ENV=production
APP_DEBUG=false
APP_URL=https://api.elearning.bpsdm.example.com

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=bpsdm_elearning_prod
DB_USERNAME=username
```

```
DB_PASSWORD=password

SANCTUM_STATEFUL_DOMAINS=api.elearning.bpsdm.example.com
SESSION_DOMAIN=.elearning.bpsdm.example.com

MAIL_MAILER=smtp
MAIL_HOST=smtp.example.com
MAIL_PORT=587
MAIL_USERNAME=username
MAIL_PASSWORD=password
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=noreply@bpsdm.example.com
MAIL_FROM_NAME="${APP_NAME}"
```

5. **Setup Database**:

bash
```
php artisan migrate --seed
```

6. **Optimasi Laravel**:

bash
```
php artisan config:cache
php artisan route:cache
php artisan storage:link
```

7. **Setup Directory Permissions**:

bash
```
chmod -R 775 storage bootstrap/cache
chown -R www-data:www-data storage bootstrap/cache
```

8. **Setup Cron Job untuk Scheduling**:

bash
```
* * * * * cd /path/to/bpsdm-elearning-api && php artisan schedule:run >> /dev/null 2>&1
```

9. **Setup Queue Worker dengan Supervisor**:
```
[program:laravel-worker]
process_name=%(program_name)s_%(process_num)02d
command=php /path/to/bpsdm-elearning-api/artisan queue:work --sleep=3 --tries=3
```

```
autostart=true
autorestart=true
user=www-data
numprocs=8
redirect_stderr=true
stdout_logfile=/var/log/supervisor/worker.log
```

Dengan mengikuti tutorial ini, Anda akan memiliki backend API yang lengkap untuk sistem e-learning BPSDM berdasarkan ERD yang diberikan. API ini dapat diintegrasikan dengan berbagai frontend seperti aplikasi web berbasis JavaScript (Vue.js, React, Angular), aplikasi mobile (React Native, Flutter), atau aplikasi desktop.