LAMPIRAN

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function (Blueprint
$table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

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

<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('sejarahs', function (Blueprint $table) {
            $table->id();
            $table->string('judul');
            $table->string('penulis');
            $table->text('isi');
            $table->string('foto')->nullable();
            $table->string('video')->nullable();
            $table->timestamps();
        });
```

```
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('sejarahs');
    }
};
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('budayas', function (Blueprint $table) {
            $table->id();
            $table->string('judul');
            $table->string('penulis');
            $table->text('isi');
            $table->string('foto')->nullable();
            $table->string('video')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('budayas');
    }
};
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('tokohs', function (Blueprint $table) {
            $table->id();
            $table->string('judul');
            $table->string('penulis');
            $table->text('isi');
            $table->string('foto')->nullable();
            $table->string('video')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('tokohs');
    }
};
<?php
```

```
namespace App\Models;

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

class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable,HasApiTokens;

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

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Tokoh extends Model
{
        protected $fillable = [
        'judul',
        'penulis',
        'isi',
        'foto',
        'video'
    ];
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Sejarah extends Model
{
    protected $fillable = [
        'judul',
        'penulis',
        'isi',
        'foto',
```

```
        'video'
    ];
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Budaya extends Model
{
        protected $fillable = [
        'judul',
        'penulis',
        'isi',
        'foto',
        'video'
    ];
}

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;

class RegisteredUserController extends Controller
{
    /**
     * Handle an incoming registration request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): Response
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255',
'unique:'.User::class],
            'password' => ['required', 'confirmed',
Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->string('password')),
        ]);

        event(new Registered($user));

        Auth::login($user);

        return response()->noContent();
    }
}
<?php

namespace App\Http\Controllers;

use App\Models\Budaya;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class BudayaController extends Controller
{
    public function index()
    {
```

```
        // Ambil dari cache jika ada, jika tidak ambil dari DB dan cache selama 60 menit
        $budaya = Cache::remember('budaya_all_data', 60, function () {
            return Budaya::all();
        });

        return response()->json($budaya, 200);
    }

    public function store(Request $request)
    {
        $request->validate([
            'judul' => 'required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->all();

        if ($request->hasFile('foto')) {
            $data['foto'] = $request->file('foto')->store('budaya/foto', 'public');
        }

        if ($request->hasFile('video')) {
            $data['video'] = $request->file('video')->store('budaya/video', 'public');
        }

        $budaya = Budaya::create($data);

        // Hapus cache karena data berubah
        Cache::forget('budaya_all_data');

        return response()->json($budaya, 201);
    }

    public function show(Budaya $budaya)
    {
        return response()->json($budaya, 200);
    }

    public function update(Request $request, Budaya $budaya)
    {
        $request->validate([
            'judul' => 'sometimes|required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->except(['foto', 'video']);

        if ($request->hasFile('foto')) {
            if ($budaya->foto) Storage::disk('public')->delete($budaya->foto);
            $data['foto'] = $request->file('foto')->store('uploads/foto', 'public');
        }

        if ($request->hasFile('video')) {
            if ($budaya->video) Storage::disk('public')->delete($budaya->video);
            $data['video'] = $request->file('video')->store('uploads/video', 'public');
        }

        $budaya->update($data);

        // Hapus cache karena data berubah
        Cache::forget('budaya_all_data');

        return response()->json($budaya, 200);
    }

    public function destroy(Budaya $budaya)
```

```
    {
        if (!empty($budaya->foto) && Storage::disk('public')->exists($budaya->foto)) {
            Storage::disk('public')->delete($budaya->foto);
        }

        if (!empty($budaya->video) && Storage::disk('public')->exists($budaya->video)) {
            Storage::disk('public')->delete($budaya->video);
        }

        $budaya->delete();

        // Hapus cache karena data dihapus
        Cache::forget('budaya_all_data');

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
}
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;
use Illuminate\Support\Facades\Auth;
class ProfileController extends Controller
{
    /**
     * Update the authenticated user's profile.
     */
    public function update(Request $request)
    {
        $user = $request->user();

        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'email', 'max:255', Rule::unique('users')->ignore($user->id)],
            'password' => ['nullable', 'string', 'min:8', 'confirmed'], // use password_confirmation
        ]);

        $user->name = $validated['name'];
        $user->email = $validated['email'];
        if (!empty($validated['password'])) {
            $user->password = Hash::make($validated['password']);
        }

        $user->save();

        return response()->json([
            'message' => 'Profile updated successfully.',
            'user' => $user,
        ]);
    }


}

<?php

namespace App\Http\Controllers;

use App\Models\Sejarah;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class SejarahController extends Controller
{
    public function index()
    {
        // Ambil data dari cache jika ada, jika tidak query dan simpan selama 60 menit
```

```
        $sejarah = Cache::remember('sejarah_all_data', 60, function ()
{
            return Sejarah::all();
        });

        return response()->json($sejarah, 200);
    }

    public function store(Request $request)
    {
        $request->validate([
            'judul' => 'required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->all();

        if ($request->hasFile('foto')) {
            $data['foto'] = $request->file('foto')->store('sejarah/foto',
'public');
        }

        if ($request->hasFile('video')) {
            $data['video'] = $request->file('video')-
>store('sejarah/video', 'public');
        }

        $sejarah = Sejarah::create($data);

        // Hapus cache karena data baru ditambahkan
        Cache::forget('sejarah_all_data');

        return response()->json($sejarah, 201);
    }

    public function show(Sejarah $sejarah)
    {
        return response()->json($sejarah, 200);
    }

    public function update(Request $request, Sejarah $sejarah)
    {
        $request->validate([
            'judul' => 'sometimes|required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->except(['foto', 'video']);

        if ($request->hasFile('foto')) {
            if ($sejarah->foto) Storage::disk('public')->delete($sejarah-
>foto);
            $data['foto'] = $request->file('foto')->store('uploads/foto',
'public');
        }

        if ($request->hasFile('video')) {
            if ($sejarah->video) Storage::disk('public')-
>delete($sejarah->video);
            $data['video'] = $request->file('video')-
>store('uploads/video', 'public');
        }

        $sejarah->update($data);

        // Hapus cache karena data diperbarui
        Cache::forget('sejarah_all_data');

        return response()->json($sejarah, 200);
    }

    public function destroy(Sejarah $sejarah)
    {
```

```
        if (!empty($sejarah->foto) && Storage::disk('public')->exists($sejarah->foto)) {
            Storage::disk('public')->delete($sejarah->foto);
        }

        if (!empty($sejarah->video) && Storage::disk('public')->exists($sejarah->video)) {
            Storage::disk('public')->delete($sejarah->video);
        }

        $sejarah->delete();

        // Hapus cache karena data dihapus
        Cache::forget('sejarah_all_data');

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Tokoh;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Cache;

class TokohController extends Controller
{
    public function index()
    {
        // Cache data semua tokoh selama 60 menit
        $tokoh = Cache::remember('tokoh_all_data', 60, function () {
            return Tokoh::all();
        });
        return response()->json($tokoh, 200);
    }

    public function store(Request $request)
    {
        $request->validate([
            'judul' => 'required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->all();

        if ($request->hasFile('foto')) {
            $data['foto'] = $request->file('foto')->store('tokoh/foto', 'public');
        }

        if ($request->hasFile('video')) {
            $data['video'] = $request->file('video')->store('tokoh/video', 'public');
        }

        $tokoh = Tokoh::create($data);

        // Hapus cache karena data baru ditambahkan
        Cache::forget('tokoh_all_data');

        return response()->json($tokoh, 201);
    }

    public function show(Tokoh $tokoh)
    {
        return response()->json($tokoh, 200);
    }

    public function update(Request $request, Tokoh $tokoh)
```

```
    {
        $request->validate([
            'judul' => 'sometimes|required|string|max:255',
            'penulis' => 'required|string',
            'isi' => 'required',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg|max:10048',
            'video' => 'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        $data = $request->except(['foto', 'video']);

        if ($request->hasFile('foto')) {
            if ($tokoh->foto) Storage::disk('public')->delete($tokoh->foto);
            $data['foto'] = $request->file('foto')->store('uploads/foto', 'public');
        }

        if ($request->hasFile('video')) {
            if ($tokoh->video) Storage::disk('public')->delete($tokoh->video);
            $data['video'] = $request->file('video')->store('uploads/video', 'public');
        }

        $tokoh->update($data);

        // Hapus cache karena data diubah
        Cache::forget('tokoh_all_data');

        return response()->json($tokoh, 200);
    }

    public function destroy(Tokoh $tokoh)
    {
        if (!empty($tokoh->foto) && Storage::disk('public')->exists($tokoh->foto)) {
            Storage::disk('public')->delete($tokoh->foto);
        }
        if (!empty($tokoh->video) && Storage::disk('public')->exists($tokoh->video)) {
            Storage::disk('public')->delete($tokoh->video);
        }

        $tokoh->delete();

        // Hapus cache karena data dihapus
        Cache::forget('tokoh_all_data');

        return response()->json(['message' => 'Data deleted successfully'], 200);
    }
}
```

```
<?php

use App\Models\Budaya;
use App\Models\Sejarah;
use Illuminate\Http\Request;
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\SejarahController;
use App\Http\Controllers\BudayaController;
use App\Http\Controllers\TokohController;
use Illuminate\Support\Facades\Route;

Route::middleware(['auth:sanctum'])->get('/user', function (Request $request) {
    return $request->user();
});
Route::middleware('auth:sanctum')->put('/profile', [ProfileController::class, 'update']);
```

```
Route::post('/register', [RegisteredUserController::class, 'store'])
    ->middleware('guest')
    ->name('register');

Route::post('/login', [AuthenticatedSessionController::class, 'store'])
    ->middleware('guest')
    ->name('login');

Route::post('/forgot-password', [PasswordResetLinkController::class, 'store'])
    ->middleware('guest')
    ->name('password.email');

Route::post('/reset-password', [NewPasswordController::class, 'store'])
    ->middleware('guest')
    ->name('password.store');

Route::get('/verify-email/{id}/{hash}', VerifyEmailController::class)
    ->middleware(['auth', 'signed', 'throttle:6,1'])
    ->name('verification.verify');

Route::post('/email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
    ->middleware(['auth', 'throttle:6,1'])
    ->name('verification.send');

Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])
    ->middleware('auth')
    ->name('logout');

// Sejarah
Route::get('/sejarah', [SejarahController::class, 'index']);
Route::post('/sejarah', [SejarahController::class, 'store']);
Route::get('/sejarah/{sejarah}', [SejarahController::class, 'show']);
Route::post('/sejarah/{sejarah}', [SejarahController::class, 'update']);
Route::delete('/sejarah/{sejarah}', [SejarahController::class, 'destroy']);

// Budaya
Route::get('/budaya', [BudayaController::class, 'index']);
Route::post('/budaya', [BudayaController::class, 'store']);
Route::get('/budaya/{budaya}', [BudayaController::class, 'show']);
Route::post('/budaya/{budaya}', [BudayaController::class, 'update']);
Route::delete('/budaya/{budaya}', [BudayaController::class, 'destroy']);

// Tokoh
Route::get('/tokoh', [TokohController::class, 'index']);
Route::post('/tokoh', [TokohController::class, 'store']);
Route::get('/tokoh/{tokoh}', [TokohController::class, 'show']);
Route::post('/tokoh/{tokoh}', [TokohController::class, 'update']);
Route::delete('/tokoh/{tokoh}', [TokohController::class, 'destroy']);

import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';

const API_URL = 'https://sejarah.proyekkk.com/api';
// const API_URL = 'http://192.168.1.7/api'; // Jika ingin tes di lokal

// Konfigurasi instance axios
const api = axios.create({
    baseURL: API_URL,
    headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }
});

// Interceptor untuk menambahkan token otomatis di setiap request
api.interceptors.response.use(response => {
    if (response.data.token) {
```

```
      AsyncStorage.setItem('userToken', response.data.token);
      api.defaults.headers.common['Authorization'] = `Bearer ${response.data.token}`;
    }
    return response;
}, error => Promise.reject(error));


// 🔷 Registrasi
export const register = async (data) => {
    try {
        const response = await api.post('/register', data);
        return response.data;
    } catch (error) {
        throw error.response?.data || { message: 'Registrasi gagal!' };
    }
};

// 🔷 Login
export const login = async (data) => {
    try {
        console.log('🔍 Memulai proses login...', data);

        const response = await api.post('/login', data);
        const token = response.data.token;

        if (!token) throw { message: 'Token tidak diterima dari server' };

        // Simpan token ke AsyncStorage
        await AsyncStorage.setItem('userToken', token);
        console.log('🔐 Token tersimpan:', token);

        return response.data;
    } catch (error) {
        console.error('❌ Error saat login:', error.response?.data || error);
        throw error.response?.data || { message: 'Login gagal!' };
    }
};

// 🔷 Logout
export const logout = async () => {
    try {
        const token = await AsyncStorage.getItem('userToken');
        if (!token) throw { message: 'Tidak ada token untuk logout' };

        await api.post('/logout', {}, { headers: { Authorization: `Bearer ${token}` } });

        console.log('🔒 Token dihapus dari AsyncStorage');
        await AsyncStorage.removeItem('userToken'); // Hapus token setelah logout sukses
        return { message: 'Logout berhasil' };
    } catch (error) {
        throw error.response?.data || { message: 'Logout gagal!' };
    }
};

// 🔷 Mendapatkan data user
export const getUser = async (token = null) => {
    try {
        const headers = token ? { Authorization: `Bearer ${token}` } : {};
        const response = await api.get('/user', { headers });
        return response.data;
    } catch (error) {
        throw error.response?.data || { message: 'Gagal mendapatkan data user!' };
    }
};

// 🔷 Update profil user (name, email, password)
export const updateProfile = async (data) => {
    try {
        const token = await AsyncStorage.getItem('userToken');
        if (!token) throw { message: 'Token tidak tersedia' };
```

```
      const response = await api.put('/profile', data, {
        headers: {
          Authorization: `Bearer ${token}`
        }
      });

      return response.data;
    } catch (error) {
      console.error('❌ Error saat update profile:', error.response?.data || error);
      throw error.response?.data || { message: 'Gagal memperbarui profil!' };
    }
};

//api sejarah
export const getsejarah = () => api.get('/sejarah');
export const getsejarahById = (id) => api.get(`/sejarah/${id}`);
export const deletesejarah = (id) => api.delete(`/sejarah/${id}`);

export const createsejarah = async (formData) => {
  try {
    const response = await api.post('/sejarah', formData, {
      headers: {
        'Content-Type': 'multipart/form-data',
      },
    });
    console.log('Status:', response.status, 'Response:', response.data);
    return response.data;
  } catch (error) {
    console.error('Gagal mengirim data:', error.response?.data || error.message);
    throw error;
  }
};

export const updatesejarah = async (id, data) => {
  return api.post(`/sejarah/${id}?_method=POST`, data, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  });
};

//api budaya
export const getbudaya = () => api.get('/budaya');
export const getbudayaById = (id) => api.get(`/budaya/${id}`);
export const deletebudaya = (id) => api.delete(`/budaya/${id}`);

export const createbudaya = async (formData) => {
  try {
    const response = await api.post('/budaya', formData, {
      headers: {
        'Content-Type': 'multipart/form-data',
      },
    });
    console.log('Status:', response.status, 'Response:', response.data);
    return response.data;
  } catch (error) {
    console.error('Gagal mengirim data:', error.response?.data || error.message);
    throw error;
  }
};

export const updatebudaya = async (id, data) => {
  return api.post(`/budaya/${id}?_method=POST`, data, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  });
};

//api tokoh
```

```
export const gettokoh = () => api.get('/tokoh');

export const gettokohById = (id) => api.get(`/tokoh/${id}`);

export const deletetokoh = (id) => api.delete(`/tokoh/${id}`);


export const createtokoh = async (formData) => {
  try {
    const response = await api.post('/tokoh', formData, {
      headers: {
        'Content-Type': 'multipart/form-data',
      },
    });
    console.log('Status:', response.status, 'Response:', response.data);
    return response.data;
  } catch (error) {
    console.error('Gagal mengirim data:', error.response?.data ||
error.message);
    throw error;
  }
};



export const updatetokoh = async (id, data) => {
  return api.post(`/tokoh/${id}?_method=POST`, data, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  });
};


export default api;

import { enableScreens } from 'react-native-screens';

enableScreens();

import { StatusBar } from 'expo-status-bar';

import { NavigationContainer } from '@react-navigation/native';

import { createNativeStackNavigator } from '@react-
navigation/native-stack';

import { createContext, useEffect, useState } from 'react';

import { getUser } from './api';

import { View, ActivityIndicator,Text } from 'react-native';

import tw from 'twrnc';

import LoginScreen from './Page/Auth/LoginScreen';

import RegisterScreen from './Page/Auth/RegisterScreen';

import BottomNavbar from './Component/BottomNavbar';

import AsyncStorage from '@react-native-async-storage/async-
storage';

import Header from './Component/Header';

import WelcomeScreen from './Page/WelcomeScreen';

import Sejarahindex from './Page/sejarah/Sejarahindex';

import Sejarahcreate from './Page/sejarah/Sejarahcreate';

import Sejarahedit from './Page/sejarah/Sejarahedit';

import SejarahDetail from './Page/sejarah/SejarahDetail';

import Budayaindex from './Page/Budaya/Budayaindex';

import Budayacreate from './Page/Budaya/Budayacreate';

import Budayaedit from './Page/Budaya/Budayaedit';

import BudayaDetail from './Page/Budaya/BudayaDetail';

import Tokohindex from './Page/Tokoh/Tokohindex';

import Tokohcreate from './Page/Tokoh/Tokohcreate';

import Tokohedit from './Page/Tokoh/Tokohedit';

import TokohDetail from './Page/Tokoh/TokohDetail';

import Jelajah from './Page/Jelajah';

import GlobalErrorBoundary from
'./Component/GlobalErrorBoundary';

import { AuthContext } from './Component/AuthContext';



const Stack = createNativeStackNavigator();


export default function App() {
    const [user, setUser] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    const [checkingToken, setCheckingToken] = useState(true);


    const checkAuth = async () => {
      setLoading(true);
      setError(null);


      try {
```

```
        const token = await AsyncStorage.getItem('userToken');
        if (!token) {
          setUser(null);
        } else {
          const userData = await getUser();
          setUser(userData);
        }
      } catch (err) {
        setError(err.message || 'Terjadi kesalahan saat memuat data.');
        setUser(null);
      } finally {
        setLoading(false);
        setCheckingToken(false);
      }
    };

    // Gunakan useEffect hanya sekali saat aplikasi pertama kali dijalankan
    useEffect(() => {
      checkAuth();
    }, []);

    if (checkingToken) {
      return (
        <View style={tw`flex-1 justify-center items-center bg-gray-100`}>
          <ActivityIndicator size="large" color="#0000ff" />
          <Text style={tw`mt-4 text-lg text-gray-700`}>Memuat aplikasi...</Text>
        </View>
      );
    }

    return (
      <GlobalErrorBoundary>
        <AuthContext.Provider value={{ user, setUser }}>
          <NavigationContainer>
            <Header />
            <Stack.Navigator screenOptions={{ headerShown: false }}>
              {!user ? (
                <>
                  <Stack.Screen name="Welcome" component={WelcomeScreen} />
                  <Stack.Screen name="Login" component={LoginScreen} />
                  <Stack.Screen name="Register" component={RegisterScreen} />
                </>
              ) : (
                <Stack.Screen name="Main" component={BottomNavbar} />
              )}

              <Stack.Screen name="Sejarahindex" component={Sejarahindex} />
              <Stack.Screen name="Tambahsejarah" component={Sejarahcreate} />
              <Stack.Screen name="EditSejarah" component={Sejarahedit} />
              <Stack.Screen name="DetailSejarah" component={SejarahDetail} />
              <Stack.Screen name="Jelajah" component={Jelajah}/>
              <Stack.Screen name="Budayaindex" component={Budayaindex} />
              <Stack.Screen name="TambahBudaya" component={Budayacreate} />
              <Stack.Screen name="EditBudaya" component={Budayaedit} />
              <Stack.Screen name="DetailBudaya" component={BudayaDetail} />
              <Stack.Screen name="Tokohindex" component={Tokohindex} />
              <Stack.Screen name="TambahTokoh" component={Tokohcreate} />
              <Stack.Screen name="EditTokoh" component={Tokohedit} />
              <Stack.Screen name="DetailTokoh" component={TokohDetail} />
            </Stack.Navigator>
```

```
            <StatusBar style="auto" />x
          </NavigationContainer>
        </AuthContext.Provider>
      </GlobalErrorBoundary>
    );
}

import { useState, useContext } from 'react';
import { View, Text, TextInput, Alert, TouchableOpacity,
ActivityIndicator, ImageBackground } from 'react-native';
import { login } from '../../api';
import { AuthContext } from '../../Component/AuthContext';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';
import AsyncStorage from '@react-native-async-storage/async-
storage';
import { useNavigation } from '@react-navigation/native';

export default function LoginScreen() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [isLoading, setIsLoading] = useState(false);
  const { setUser } = useContext(AuthContext);
  const [showPassword, setShowPassword] = useState(false);
  const navigation = useNavigation();

  const handleLogin = async () => {
    if (!email.trim() || !password.trim()) {
      Alert.alert('Error', 'Email dan password tidak boleh
kosong!');
      return;
    }

    setIsLoading(true);
    try {
      const response = await login({ email, password });
      await AsyncStorage.setItem('userToken', response.token);
      setUser(response.user);
      Alert.alert('Berhasil', 'Login sukses!');
    } catch (error) {
      console.error('❌ Login gagal:', error);
      let message = 'Login gagal!';
      const msg = error?.message;

      if (msg === 'These credentials do not match our records.') {
        message = 'Email atau kata sandi Anda salah';
      }

      Alert.alert('Gagal', message);
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <ImageBackground
      source={require('../../assets/bg.jpg')} // gunakan
background yang sama seperti landing page
      style={tw`flex-1`}
    >
      <View style={tw`flex-1 justify-center px-6 bg-black/60`}>
        <Text style={tw`text-yellow-500 text-3xl font-bold text-
center mb-8`}>Masuk ke Akun Anda</Text>

        <TextInput
          placeholder="Email"
          placeholderTextColor="#ccc"
          value={email}
          onChangeText={setEmail}
          style={tw`bg-white rounded-xl p-4 mb-4 shadow-lg`}
          keyboardType="email-address"
          autoCapitalize="none"
        />

        <View style={tw`flex-row items-center bg-white
rounded-xl p-4 mb-4 shadow-lg`}>
          <TextInput
            placeholder="Password"
```

```
                placeholderTextColor="#999"
                value={password}
                onChangeText={setPassword}
                style={tw`flex-1`}
                secureTextEntry={!showPassword}
              />
              <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>
                <FontAwesome name={showPassword ? 'eye-slash' : 'eye'} size={20} />
              </TouchableOpacity>
            </View>

            <TouchableOpacity
              style={tw`bg-yellow-500 rounded-full py-3 items-center mb-4`}
              onPress={handleLogin}
              disabled={isLoading}
            >
              {isLoading ? (
                <ActivityIndicator color="#000" />
              ) : (
                <Text style={tw`text-black font-bold text-lg`}>Login</Text>
              )}
            </TouchableOpacity>

            {/* <TouchableOpacity onPress={() => navigation.navigate('Register')}>
              <Text style={tw`text-yellow-400 text-center mt-2`}>
                Belum punya akun? Daftar di sini
              </Text>
            </TouchableOpacity> */}
          </View>
        </ImageBackground>
    );
}import { useState } from 'react';
import {
    View,
    Text,
    TextInput,
    TouchableOpacity,
    Alert,
    ActivityIndicator,
    ImageBackground
} from 'react-native';
import { register } from '../../api';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';

export default function RegisterScreen({ navigation }) {
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [passwordConfirmation, setPasswordConfirmation] = useState('');
    const [showPassword, setShowPassword] = useState(false);
    const [showPasswordConfirmation, setShowPasswordConfirmation] = useState(false);
    const [isLoading, setIsLoading] = useState(false);

    const handleRegister = async () => {
        if (password !== passwordConfirmation) {
            Alert.alert('Gagal', 'Password dan Konfirmasi Password tidak sama!');
            return;
        }

        setIsLoading(true);
        try {
            await register({
                name,
                email,
                password,
                password_confirmation: passwordConfirmation
            });
            Alert.alert('Berhasil', 'Registrasi sukses!');
            navigation.navigate('Login');
        } catch (error) {
```

```
      Alert.alert('Gagal', error.message || 'Registrasi gagal!');
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <ImageBackground
      source={require('../../assets/bg.jpg')} // Ganti path sesuai lokasi
      style={tw`flex-1`}
    >
      <View style={tw`flex-1 justify-center items-center bg-black/60 px-6`}>
        <Text style={tw`text-white text-3xl font-bold mb-6`}>Daftar Akun</Text>

        <TextInput
          placeholder="Nama"
          placeholderTextColor="#ccc"
          value={name}
          onChangeText={setName}
          style={tw`w-full bg-white/90 text-black p-4 mb-4 rounded-full`}
        />

        <TextInput
          placeholder="Email"
          placeholderTextColor="#ccc"
          value={email}
          onChangeText={setEmail}
          style={tw`w-full bg-white/90 text-black p-4 mb-4 rounded-full`}
          keyboardType="email-address"
          autoCapitalize="none"
        />

        <View style={tw`w-full bg-white/90 flex-row items-center p-4 mb-4 rounded-full`}>
          <TextInput
            placeholder="Password"
            placeholderTextColor="#999"
            value={password}
            onChangeText={setPassword}
            style={tw`flex-1 text-black`}
            secureTextEntry={!showPassword}
          />
          <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>
            <FontAwesome name={showPassword ? "eye-slash" : "eye"} size={20} />
          </TouchableOpacity>
        </View>

        <View style={tw`w-full bg-white/90 flex-row items-center p-4 mb-6 rounded-full`}>
          <TextInput
            placeholder="Konfirmasi Password"
            placeholderTextColor="#999"
            value={passwordConfirmation}
            onChangeText={setPasswordConfirmation}
            style={tw`flex-1 text-black`}
            secureTextEntry={!showPasswordConfirmation}
          />
          <TouchableOpacity onPress={() => setShowPasswordConfirmation(!showPasswordConfirmation)}>
            <FontAwesome name={showPasswordConfirmation ? "eye-slash" : "eye"} size={20} />
          </TouchableOpacity>
        </View>

        <TouchableOpacity
          style={tw`bg-yellow-500 px-8 py-3 rounded-full w-full mb-4 items-center`}
          onPress={handleRegister}
          disabled={isLoading}
        >
          {isLoading ? (
            <ActivityIndicator color="#000" />
          ) : (
```

```
                    <Text style={tw`text-black font-bold text-lg`}>Daftar</Text>
                )}
            </TouchableOpacity>

            <TouchableOpacity onPress={() => navigation.navigate('Login')}>
                <Text style={tw`text-yellow-500 text-sm mt-2`}>Sudah punya akun? Masuk di sini</Text>
            </TouchableOpacity>
        </View>
    </ImageBackground>
  );
}
```

```
import { View, Text, TouchableOpacity, ImageBackground, ActivityIndicator } from 'react-native';
import { useEffect, useContext, useState } from 'react';
import { getUser, logout } from '../api';
import tw from 'twrnc';
import { AuthContext } from '../Component/AuthContext';

export default function DashboardScreen({ navigation }) {
  const { user, setUser } = useContext(AuthContext);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchUser = async () => {
      try {
        const userData = await getUser();
        setUser(userData);
      } catch (error) {
        setUser(null);
      } finally {
        setLoading(false);
      }
    };
    fetchUser();
  }, []);

  return (
    <ImageBackground
      source={require('../assets/bg.jpg')} // Ganti path sesuai lokasi background landing page
      style={tw`flex-1`}
    >
      <View style={tw`flex-1 bg-black/60 px-6 justify-center items-center`}>
        <Text style={tw`text-white text-3xl font-bold mb-6`}>Dashboard</Text>

        {loading ? (
          <ActivityIndicator size="large" color="#FFD700" />
        ) : user ? (
          <View style={tw`bg-white/90 p-6 rounded-2xl shadow-lg w-full mb-6`}>
            <Text style={tw`text-lg font-bold text-black mb-2`}>
              Selamat datang, {user.name}!
            </Text>
            <Text style={tw`text-sm text-gray-700`}>Email: {user.email}</Text>
          </View>
        ) : (
          <Text style={tw`text-white text-lg mb-4`}>Anda belum login</Text>
        )}

      </View>
    </ImageBackground>
  );
}
```

```
import React from 'react';
import { View, Text, TouchableOpacity, ImageBackground, ScrollView } from 'react-native';
import LottieView from 'lottie-react-native';
import tw from 'twrnc';

export default function Jelajah({ navigation }) {
  return (
```

```
      <ImageBackground
        source={require('../assets/bg.jpg')} // Ganti dengan nuansa sejarah
        style={tw`flex-1`}
      >
        <ScrollView contentContainerStyle={tw`flex-grow justify-center items-center bg-black/60 px-6 py-10`}>
          <LottieView
            source={require('../assets/sejarah.json')} // Animasikan tema sejarah
            autoPlay
            loop
            style={{ width: 250, height: 250 }}
          />

          <Text style={tw`text-white text-3xl font-bold text-center mt-4`}>
            Jelajahi Sejarah Daerahmu
          </Text>

          <Text style={tw`text-gray-200 text-center mt-2 mb-8`}>
            Temukan kisah, budaya, dan para pahlawan yang membentuk jati diri daerah kita.
          </Text>

          {/* Menu 1: Sejarah */}
          <TouchableOpacity
            style={tw`bg-yellow-500 px-10 py-4 rounded-full mb-4 w-full`}
            onPress={() => navigation.navigate('Sejarahindex')}
          >
            <Text style={tw`text-black text-lg font-bold text-center`}>Sejarah</Text>
          </TouchableOpacity>

          {/* Menu 2: Budaya */}
          <TouchableOpacity
            style={tw`bg-white px-10 py-4 rounded-full mb-4 w-full border border-yellow-500`}
            onPress={() => navigation.navigate('Budayaindex')}
          >
            <Text style={tw`text-yellow-500 text-lg font-bold text-center`}>Budaya</Text>
          </TouchableOpacity>

          {/* Menu 3: Pahlawan Lokal */}
          <TouchableOpacity
            style={tw`bg-transparent border border-white px-10 py-4 rounded-full w-full`}
            onPress={() => navigation.navigate('Tokohindex')}
          >
            <Text style={tw`text-white text-lg font-bold text-center`}>Pahlawan Lokal</Text>
          </TouchableOpacity>
        </ScrollView>
      </ImageBackground>
    );
}
import React, { useEffect, useState, useContext } from 'react';
import {
  View,
  Text,
  ScrollView,
  ActivityIndicator,
  TextInput,
  TouchableOpacity,
  Alert,
  ImageBackground,
  Image
} from 'react-native';
import tw from 'twrnc';
import { gettokoh, deletetokoh } from '../../api';
import { AuthContext } from "../../Component/AuthContext";

const Tokohindex = ({ navigation }) => {
  const { user } = useContext(AuthContext) || { user: null };
  const [tokoh, setTokoh] = useState([]);
  const [filteredTokoh, setFilteredTokoh] = useState([]);
  const [search, setSearch] = useState('');
  const [loading, setLoading] = useState(true);
```

```
useEffect(() => {
  fetchTokoh();
}, []);

useEffect(() => {
  handleSearch(search);
}, [search, tokoh]);

const fetchTokoh = async () => {
  setLoading(true);
  try {
    const response = await gettokoh();
    setTokoh(response.data);
    setFilteredTokoh(response.data);
  } catch (error) {
    console.error('Gagal memuat data Tokoh:', error);
  } finally {
    setLoading(false);
  }
};

const handleSearch = (text) => {
  setSearch(text);
  const filtered = tokoh.filter((item) =>
    item.judul.toLowerCase().includes(text.toLowerCase())
  );
  setFilteredTokoh(filtered);
};

const handleDelete = (id) => {
  Alert.alert(
    'Konfirmasi Hapus',
    'Apakah Anda yakin ingin menghapus item ini?',
    [
      { text: 'Batal', style: 'cancel' },
      {
        text: 'Hapus',
        style: 'destructive',
        onPress: async () => {
          try {
            await deletetokoh(id);
            fetchTokoh();
          } catch (error) {
            console.error('Gagal menghapus:', error);
            Alert.alert('Error', 'Gagal menghapus data.');
          }
        },
      },
    ]
  );
};

  if (loading) {
    return (
      <View style={tw`flex-1 justify-center items-center bg-black`}>
        <ActivityIndicator size="large" color="#FACC15" />
        <Text style={tw`mt-4 text-yellow-400`}>Memuat data
Tokoh...</Text>
      </View>
    );
  }

  return (
    <ImageBackground
      source={require('../../assets/bg.jpg')}
      style={tw`flex-1`}
    >
      <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
        <Text style={tw`text-2xl font-bold text-yellow-400 text-
center mb-6`}>
          Daftar Tokoh
        </Text>

        {user && (
          <TouchableOpacity
            style={tw`bg-yellow-500 py-3 rounded-full mb-4`}
```

```
          onPress={() => navigation.navigate("TambahTokoh")}
        >
          <Text style={tw`text-black text-center font-bold`}>
            + Tambah Tokoh
          </Text>
        </TouchableOpacity>
      )}

      <TextInput
        placeholder="Cari berdasarkan judul..."
        value={search}
        onChangeText={handleSearch}
        style={tw`bg-white/80 border border-yellow-400 rounded-xl
px-4 py-2 mb-4 text-gray-900`}
        placeholderTextColor="#6B7280"
      />

      <ScrollView>
        {filteredTokoh.length > 0 ? (
          filteredTokoh.map((item, index) => (
            <View
              key={index}
              style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
            >
              <Text style={tw`text-lg font-semibold text-black`}>
                {item.judul}
              </Text>
              {item?.foto && (
                <Image
                  source={{ uri:
`https://sejarah.proyekkk.com/storage/${item.foto}` }}
                  style={tw`w-full h-60 rounded-xl mb-6`}
                  resizeMode="cover"
                />
              )}

              <Text style={tw`mt-1 text-gray-700`}>
                {item.isi.split(' ').length > 5
                  ? item.isi.split(' ').slice(0, 5).join(' ') + '...'
                  : item.isi}
              </Text>

              <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
                {user && (
                  <>
                    <TouchableOpacity
                      onPress={() => handleDelete(item.id)}
                      style={tw`bg-red-600 px-4 py-2 rounded-full`}
                    >
                      <Text style={tw`text-white font-
semibold`}>Hapus</Text>
                    </TouchableOpacity>

                    <TouchableOpacity
                      onPress={() =>
                        navigation.navigate("EditTokoh", { id: item.id })
                      }
                      style={tw`bg-blue-600 px-4 py-2 rounded-full`}
                    >
                      <Text style={tw`text-white font-
semibold`}>Edit</Text>
                    </TouchableOpacity>
                  </>
                )}
                <TouchableOpacity
                  onPress={() =>
                    navigation.navigate("DetailTokoh", { id: item.id })
                  }
                  style={tw`bg-green-600 px-4 py-2 rounded-full`}
                >
                  <Text style={tw`text-white font-
semibold`}>Detail</Text>
                </TouchableOpacity>
              </View>
            </View>
          ))
        ) : (
          <Text style={tw`text-center text-yellow-200`}>
```

```
            Tidak ada data Tokoh yang cocok.
          </Text>
        )}
      </ScrollView>
    </View>
  </ImageBackground>
 );
};

export default Tokohindex;
import React, { useEffect, useState, useContext } from 'react';
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
 TextInput,
 TouchableOpacity,
 Alert,
 ImageBackground,
 Image
} from 'react-native';
import tw from 'twrnc';
import { getsejarah, deletesejarah } from '../../api';
import { AuthContext } from "../../Component/AuthContext";

const Sejarahindex = ({ navigation }) => {
 const { user } = useContext(AuthContext) || { user: null };
 const [sejarah, setSejarah] = useState([]);
 const [filteredSejarah, setFilteredSejarah] = useState([]);
 const [search, setSearch] = useState('');
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  fetchSejarah();
 }, []);

 useEffect(() => {
  handleSearch(search);
 }, [search, sejarah]);

 const fetchSejarah = async () => {
  setLoading(true);
  try {
   const response = await getsejarah();
   setSejarah(response.data);
   setFilteredSejarah(response.data);
  } catch (error) {
   console.error('Gagal memuat data sejarah:', error);
  } finally {
   setLoading(false);
  }
 };

 const handleSearch = (text) => {
  setSearch(text);
  const filtered = sejarah.filter((item) =>
   item.judul.toLowerCase().includes(text.toLowerCase())
  );
  setFilteredSejarah(filtered);
 };

 const handleDelete = (id) => {
  Alert.alert(
   'Konfirmasi Hapus',
   'Apakah Anda yakin ingin menghapus item ini?',
   [
    { text: 'Batal', style: 'cancel' },
    {
     text: 'Hapus',
     style: 'destructive',
     onPress: async () => {
      try {
       await deletesejarah(id);
       fetchSejarah();
      } catch (error) {
```

```
            console.error('Gagal menghapus:', error);
            Alert.alert('Error', 'Gagal menghapus data.');
          }
        },
      },
    ]
  );
};

if (loading) {
  return (
    <View style={tw`flex-1 justify-center items-center bg-black`}>
      <ActivityIndicator size="large" color="#FACC15" />
      <Text style={tw`mt-4 text-yellow-400`}>Memuat data sejarah...</Text>
    </View>
  );
}

return (
  <ImageBackground
    source={require('../../assets/bg.jpg')}
    style={tw`flex-1`}
  >
    <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
      <Text style={tw`text-2xl font-bold text-yellow-400 text-center mb-6`}>
        Daftar Sejarah
      </Text>

      {user && (
        <TouchableOpacity
          style={tw`bg-yellow-500 py-3 rounded-full mb-4`}
          onPress={() => navigation.navigate("Tambahsejarah")}
        >
          <Text style={tw`text-black text-center font-bold`}>
            + Tambah Sejarah
          </Text>
        </TouchableOpacity>
      )}

      <TextInput
        placeholder="Cari berdasarkan judul..."
        value={search}
        onChangeText={handleSearch}
        style={tw`bg-white/80 border border-yellow-400 rounded-xl px-4 py-2 mb-4 text-gray-900`}
        placeholderTextColor="#6B7280"
      />

      <ScrollView>
        {filteredSejarah.length > 0 ? (
          filteredSejarah.map((item, index) => (
            <View
              key={index}
              style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
            >
              <Text style={tw`text-lg font-semibold text-black`}>
                {item.judul}
              </Text>
                        {item?.foto && (
                <Image
                  source={{ uri: `https://sejarah.proyekkk.com/storage/${item.foto}` }}
                  style={tw`w-full h-60 rounded-xl mb-6`}
                  resizeMode="cover"
                />
              )}

              <Text style={tw`mt-1 text-gray-700`}>
{item.isi.split(' ').length > 5
  ? item.isi.split(' ').slice(0, 5).join(' ') + '...'
  : item.isi}
</Text>

              <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
                {user && (
```

```
              <>
                <TouchableOpacity
                  onPress={() => handleDelete(item.id)}
                  style={tw`bg-red-600 px-4 py-2 rounded-full`}
                >
                  <Text style={tw`text-white font-
semibold`}>Hapus</Text>
                </TouchableOpacity>

                <TouchableOpacity
                  onPress={() =>
                    navigation.navigate("EditSejarah", { id: item.id })
                  }
                  style={tw`bg-blue-600 px-4 py-2 rounded-full`}
                >
                  <Text style={tw`text-white font-
semibold`}>Edit</Text>
                </TouchableOpacity>
              </>
            )}
            <TouchableOpacity
              onPress={() =>
                navigation.navigate("DetailSejarah", { id: item.id })
              }
              style={tw`bg-green-600 px-4 py-2 rounded-full`}
            >
              <Text style={tw`text-white font-
semibold`}>Detail</Text>
            </TouchableOpacity>
          </View>
        </View>
      ))
    ) : (
      <Text style={tw`text-center text-yellow-200`}>
        Tidak ada data sejarah yang cocok.
      </Text>
    )}
  </ScrollView>
</View>
  </ImageBackground>
 );
};

export default Sejarahindex;
import React, { useEffect, useState, useContext } from 'react';
import {
 View,
 Text,
 ScrollView,
 ActivityIndicator,
 TextInput,
 TouchableOpacity,
 Alert,
 ImageBackground,
 Image
} from 'react-native';
import tw from 'twrnc';
import { getbudaya, deletebudaya } from '../../api';
import { AuthContext } from "../../Component/AuthContext";

const Budayaindex = ({ navigation }) => {
 const { user } = useContext(AuthContext) || { user: null };
 const [budaya, setbudaya] = useState([]);
 const [filteredBudaya, setFilteredBudaya] = useState([]);
 const [search, setSearch] = useState('');
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  fetchBudaya();
 }, []);

 useEffect(() => {
  handleSearch(search);
 }, [search, budaya]);

 const fetchBudaya = async () => {
  setLoading(true);
```

```
  try {
    const response = await getbudaya();
    setbudaya(response.data);
    setFilteredBudaya(response.data);
  } catch (error) {
    console.error('Gagal memuat data budaya:', error);
  } finally {
    setLoading(false);
  }
};

const handleSearch = (text) => {
  setSearch(text);
  const filtered = budaya.filter((item) =>
    item.judul.toLowerCase().includes(text.toLowerCase())
  );
  setFilteredBudaya(filtered);
};

const handleDelete = (id) => {
  Alert.alert(
    'Konfirmasi Hapus',
    'Apakah Anda yakin ingin menghapus item ini?',
    [
      { text: 'Batal', style: 'cancel' },
      {
        text: 'Hapus',
        style: 'destructive',
        onPress: async () => {
          try {
            await deletebudaya(id);
            fetchBudaya();
          } catch (error) {
            console.error('Gagal menghapus:', error);
            Alert.alert('Error', 'Gagal menghapus data.');
          }
        },
      },
    ]
  );
};

  if (loading) {
    return (
      <View style={tw`flex-1 justify-center items-center bg-black`}>
        <ActivityIndicator size="large" color="#FACC15" />
        <Text style={tw`mt-4 text-yellow-400`}>Memuat data
budaya...</Text>
      </View>
    );
  }

  return (
    <ImageBackground
      source={require('../../assets/bg.jpg')}
      style={tw`flex-1`}
    >
      <View style={tw`flex-1 bg-black/60 px-4 py-6`}>
        <Text style={tw`text-2xl font-bold text-yellow-400 text-
center mb-6`}>
          Daftar budaya
        </Text>

        {user && (
          <TouchableOpacity
            style={tw`bg-yellow-500 py-3 rounded-full mb-4`}
            onPress={() => navigation.navigate("TambahBudaya")}
          >
            <Text style={tw`text-black text-center font-bold`}>
              + Tambah budaya
            </Text>
          </TouchableOpacity>
        )}

        <TextInput
          placeholder="Cari berdasarkan judul..."
          value={search}
```

```
        onChangeText={handleSearch}
        style={tw`bg-white/80 border border-yellow-400 rounded-xl
px-4 py-2 mb-4 text-gray-900`}
        placeholderTextColor="#6B7280"
      />

      <ScrollView>
        {filteredBudaya.length > 0 ? (
          filteredBudaya.map((item, index) => (
            <View
              key={index}
              style={tw`mb-4 p-4 bg-white/80 rounded-xl shadow`}
            >
              <Text style={tw`text-lg font-semibold text-black`}>
                {item.judul}
              </Text>
                        {item?.foto && (
                <Image
                  source={{ uri:
`https://sejarah.proyekkk.com/storage/${item.foto}` }}
                  style={tw`w-full h-60 rounded-xl mb-6`}
                  resizeMode="cover"
                />
              )}

              <Text style={tw`mt-1 text-gray-700`}>
  {item.isi.split(' ').length > 5
    ? item.isi.split(' ').slice(0, 5).join(' ') + '...'
    : item.isi}
</Text>

              <View style={tw`flex-row gap-4 mt-3 flex-wrap`}>
                {user && (
                  <>
                    <TouchableOpacity
                      onPress={() => handleDelete(item.id)}
                      style={tw`bg-red-600 px-4 py-2 rounded-full`}
                    >
                      <Text style={tw`text-white font-
semibold`}>Hapus</Text>
                    </TouchableOpacity>

                    <TouchableOpacity
                      onPress={() =>
                        navigation.navigate("EditBudaya", { id: item.id })
                      }
                      style={tw`bg-blue-600 px-4 py-2 rounded-full`}
                    >
                      <Text style={tw`text-white font-
semibold`}>Edit</Text>
                    </TouchableOpacity>
                  </>
                )}
                <TouchableOpacity
                  onPress={() =>
                    navigation.navigate("DetailBudaya", { id: item.id })
                  }
                  style={tw`bg-green-600 px-4 py-2 rounded-full`}
                >
                  <Text style={tw`text-white font-
semibold`}>Detail</Text>
                </TouchableOpacity>
              </View>
            </View>
          ))
        ) : (
          <Text style={tw`text-center text-yellow-200`}>
            Tidak ada data budaya yang cocok.
          </Text>
        )}
      </ScrollView>
    </View>
  </ImageBackground>
 );
};

export default Budayaindex;
```