<?php

namespace App\Http\Controllers;

use Inertia\Response;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Log;

use Illuminate\Support\Facades\File;

use Illuminate\Http\RedirectResponse;

use Symfony\Component\HttpFoundation\BinaryFileResponse;

class BackupController extends Controller

{

    protected string $backupPath;

    public function \_\_construct()

    {

        $this->backupPath = storage\_path('app/backups');

    }

    /\*\*

     \* Display all available backup files.

     \*/

    public function index(): Response

    {

        if (!File::exists($this->backupPath)) {

            File::makeDirectory($this->backupPath, 0755, true);

        }

        $files = collect(File::files($this->backupPath))

            ->map(fn($file, $key) => [

                'id' => $key,

                'name' => $file->getFilename(),

                'size' => round($file->getSize() / 1024 / 1024, 2) . ' MB',

                'date' => date('Y-m-d H:i:s', $file->getMTime()),

            ])

            ->sortByDesc('date')

            ->values();

        return inertia('backups/index', ['files' => $files]);

    }

    /\*\*

     \* Generate a new backup file without forcing download.

     \*/

    public function generate(): RedirectResponse

    {

        $db = config('database.connections.mysql.database');

        $filename = $db . '\_backup\_' . now()->format('Y\_m\_d\_His') . '.sql';

        $file = $this->backupPath . '/' . $filename;

        // Ensure backup folder exists

        if (!File::exists($this->backupPath)) {

            File::makeDirectory($this->backupPath, 0755, true);

        }

        // Create temporary MySQL credentials file

        $configFile = $this->backupPath . '/.my.cnf';

        file\_put\_contents($configFile, "[client]\nuser=" . config('database.connections.mysql.username') . "\npassword=" . config('database.connections.mysql.password') . "\nhost=" . config('database.connections.mysql.host'));

        chmod($configFile, 0600);

        // Run mysqldump command

        $command = sprintf(

            'mysqldump --defaults-extra-file=%s %s > %s',

            escapeshellarg($configFile),

            escapeshellarg($db),

            escapeshellarg($file)

        );

        exec($command, $output, $result);

        unlink($configFile);

        if ($result !== 0) {

            Log::error('Backup generation failed', ['output' => $output]);

            return back()->with('error', 'Failed to generate backup.');

        }

        return back()->with('success', "Backup file {$filename} created successfully.");

    }

    /\*\*

     \* Download a specific existing backup.

     \*/

    public function download(Request $request): BinaryFileResponse|RedirectResponse

    {

        $file = $request->input('file');

        $path = $this->backupPath . '/' . basename($file); // sanitize filename

        if (!File::exists($path)) {

            return back()->with('error', 'Backup file not found.');

        }

        return response()->download($path);

    }

    /\*\*

     \* Delete a selected backup file.

     \*/

    public function destroy(Request $request): RedirectResponse

    {

        $file = $request->input('file');

        $path = $this->backupPath . '/' . basename($file); // sanitize filename

        if (!File::exists($path)) {

            return back()->with('error', 'Backup file not found.');

        }

        File::delete($path);

        return back()->with('success', 'Backup deleted successfully.');

    }

}