Migration

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration {

    /\*\*

     \* Run the migrations.

     \*/

    public function up(): void

    {

        Schema::create('students', function (Blueprint $table) {

            $table->id();

            $table->string('student\_number')->unique();

            $table->string('first\_name');

            $table->string('last\_name');

            $table->string('middle\_name');

            $table->timestamp('date\_of\_registration');

            $table->timestamps();

            // Add FULLTEXT index for fast search

            $table->fullText(['student\_number', 'first\_name', 'last\_name'], 'students\_fulltext\_idx');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::dropIfExists('students');

    }

};

Fetching data

<?php

namespace App\Services;

use App\Models\Student;

use Illuminate\Support\Collection;

class StudentService

{

    /\*\*

     \* @param string $search

     \* @return Collection<int, Student>

     \*/

    public function getStudents(string $search): Collection

    {

        $students = Student::query()

            ->select([

                'id',

                'student\_number',

                'first\_name',

                'last\_name',

                'date\_of\_registration',

            ])

            ->when($search, function ($query, $search) {

                    $query->whereRaw(

                        "MATCH(student\_number,first\_name, last\_name) AGAINST(? IN BOOLEAN MODE)",

                        [$search . '\*']

                    );

            })

            ->latest('id')

            ->limit(50)

            ->get();

        return $students;

    }

}

Or when sorting

<?php

use Illuminate\Support\Facades\DB;

    /\*\*

     \* @param string $search

     \* @return Collection<int, Student>

     \*/

    public function getStudents(string $search): Collection

    {

        $students = Student::query()

            ->with(['media'])

            ->select([

                'id',

                'student\_number',

                'first\_name',

                'last\_name',

                'date\_of\_registration',

            ])

            ->when($search, function ($query) use ($search) {

                $query->addSelect(DB::raw("MATCH(student\_number, first\_name, last\_name) AGAINST(? IN BOOLEAN MODE) as relevance"))

                    ->whereRaw(

                        "MATCH(student\_number, first\_name, last\_name) AGAINST(? IN BOOLEAN MODE)",

                        [$search . '\*', $search . '\*']

                    )->orderByDesc('relevance');

            });

            if(!$search){

                $students->orderBy('last\_name');

            }

        return $students->limit(50)->get();

    }