<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

use App\Models\Search;

use Illuminate\Http\Request;

use Illuminate\Database\Eloquent\ModelNotFoundException;

use App\Helpers\ErrorHelper;

use Exception;

use Illuminate\Validation\ValidationException;

class SearchController extends Controller

{

    public function select\_user($user\_search\_id){

        try {

            request()->validate(

                [

                'age'=>['required'],

                'age2'=>['required']

                ],

                [

                    'age.required'=>['پر کردن فیلد سن ضروری است']

                ]

           );

            $search = new Search();

            $search->user = request()->user()->id ;

            $search->user\_search = $user\_search\_id ;

            $search->date = now();

            $search->save();

            return response()->json(['success'=>true,'return\_user\_id'=>$user\_search\_id],200);

        }

        catch (ValidationException $e) {

            // Handle validation errors

            return response()->json([

                'message' => 'خطای اعتبارسنجی اتفاق افتاده است. لطفا ورودی ها را بررسی کنید',

                'errors' => $e->errors(),

            ], 422);

        }

        catch (Exception $e) {

            // Handle any other exceptions

            return ErrorHelper::handleError($e);

        }

        // catch (ModelNotFoundException $e) {

        //     return response()->json(['success' => false, 'message' => 'User not found.'], 404);

        // } catch (\Exception $e) {

        //     // Handle any other exceptions

        //     return response()->json(['success' => false, 'message' => 'خطایی اتفاق افتاده است'], 500);

        // }

    }

}

App\Helpers:

<?php

namespace App\Helpers;

use Exception;

use Illuminate\Database\QueryException;

use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

use Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;

use Illuminate\Http\JsonResponse;

class ErrorHelper

{

    public static function handleError(Exception $e)

    {

        // Default user-friendly message

        $userFriendlyMessage = 'An unexpected error occurred. Please try again later.';

        $detailedErrorMessage = $e->getMessage(); // Detailed error message

        // Check for specific exception types

        if ($e instanceof QueryException) {

            $errorCode = $e->errorInfo[1];

            switch ($errorCode) {

                case 1146: // Table not found

                    $userFriendlyMessage = 'We encountered an issue while trying to access our data. Please try again later.';

                    break;

                case 23000: // Integrity constraint violation

                    $userFriendlyMessage = 'It seems there is a conflict with the data you are trying to save.';

                    break;

                default:

                    $userFriendlyMessage = 'An error occurred while accessing the database. Please try again later.';

                    break;

            }

        } elseif ($e instanceof NotFoundHttpException) {

            $userFriendlyMessage = 'The resource you are looking for was not found.';

            return self::prepareResponse($userFriendlyMessage, 404, $detailedErrorMessage);

        } elseif ($e instanceof AccessDeniedHttpException) {

            $userFriendlyMessage = 'You do not have permission to access this resource.';

            return self::prepareResponse($userFriendlyMessage, 403, $detailedErrorMessage);

        }

        // Handle generic errors

        return self::prepareResponse($userFriendlyMessage, 500, $detailedErrorMessage);

    }

    private static function prepareResponse($userFriendlyMessage, $statusCode, $detailedErrorMessage)

    {

        // Check if the request expects a JSON response

        if (request()->wantsJson()) {

            // JSON response

            return response()->json([

                'message' => $userFriendlyMessage,

                'error' => $detailedErrorMessage,

            ], $statusCode);

        } else {

            // Ordinary response

            return response($userFriendlyMessage, $statusCode);

        }

    }

}

<template>

    <div>

            <div class="my-3 d-inline-block">

                <label for="title" class="form-label  d-inline-block ms-3">عنوان  آیتم : </label>

                <input type="text" class="form-control d-inline-block" id="title" v-model="form.title" style="width: 400px;">

                <div v-if="form.errors.title" class="color\_red">{{ form.errors.title }}</div>

            </div>

            <div class="mb-3 d-inline-block">

                <div for="title" class="form-label ms-2 me-5" style="display: inline-block;">نوع : </div>

                 <select v-model="form.type" class="form-control d-inline-block" style="width: 100px;" >

                     <option value="number"> عددی </option>

                     <option value="tick"> تیک </option>

                 </select>

                <div v-if="form.errors.type" class="color\_red">{{ form.errors.type }}</div>

            </div>

             <div>

                    <div class="mb-3 d-inline-block">

                      <div for="title" class="form-label ms-2 me-5" style="display: inline-block;">حداقل مقدار : </div>

                      <input type="text" class="form-control d-inline-block" id="min" v-model="form.min" style="width: 150px;">

                      <div v-if="form.errors.min" class="color\_red">{{ form.errors.min }}</div>

                  </div>

                  <div class="mb-3 d-inline-block">

                      <div for="title" class="form-label ms-2 me-5" style="display: inline-block;">حداکثر مقدار : </div>

                      <input type="text" class="form-control d-inline-block" id="min" v-model="form.max" style="width: 150px;">

                      <div v-if="form.errors.max" class="color\_red">{{ form.errors.max }}</div>

                  </div>

             </div>

            <div class="mb-3  d-inline-block">

                <div for="title" class="form-label d-inline-block" >نام  وارد کننده :  </div>

                <SearchName   :show="showSearchBox.adder" @close\_search="close\_seach\_fn"  @selectRowEmit="(res)=>updaterResults(res,'adder')" @clickSearchIcon="showSearchBox.adder=true"

                   class="d-inline-block" />

                <div v-if="form.errors.adder\_id" class="color\_red">{{ form.errors.adder\_id }}</div>

            </div>

            <div class="mb-3  d-inline-block me-5">

                <div for="title" class="form-label d-inline-block" >نام    تایید کننده :  </div>

                <SearchName   :show="showSearchBox.verifier" @close\_search="close\_seach\_fn"  @selectRowEmit="(res)=>updaterResults(res,'verifier')" @clickSearchIcon="showSearchBox.verifier=true"

                   class="d-inline-block" />

                <div v-if="form.errors.verifier\_id" class="color\_red">{{ form.errors.verifier\_id }}</div>

            </div>

            <div class="mb-3  d-inline-block me-5">

                <div for="title" class="form-label d-inline-block" >نام    مشاهده کننده :  </div>

                <SearchName   :show="showSearchBox.viewer" @close\_search="close\_seach\_fn"  @selectRowEmit="(res)=>updaterResults(res,'viewer')" @clickSearchIcon="showSearchBox.viewer=true"

                   class="d-inline-block" />

                <div v-if="form.errors.viewer\_id" class="color\_red">{{ form.errors.viewer\_id }}</div>

            </div>

            <div v-if="form.errors">

                    <div v-for="error in form.errors" class="color\_red"> {{ error }} </div>

            </div>

            <div class=" mt-2 mb-3 me-5" style="display: flex;  gap: 50px;">

              <button class="btn btn-success btn-sm " @click="sendData">ذخیره</button>

              <button class="btn btn-secondary btn-sm mx-3" @click="emit('toggle\_show')">کنسل</button>

            </div>

    </div>

  </template>

  <script setup>

  import { reactive, ref, watchEffect } from 'vue';

  import { useForm } from '@inertiajs/vue3';

  import \_ from 'lodash'

  import axios from 'axios'

  import { toast } from 'vue3-toastify'

  import 'vue3-toastify/dist/index.css';

//   import Modal from './Modal.vue';

const props = defineProps(['menu'])

const emit = defineEmits(['toggle\_show'])

  import { Modal } from 'usemodal-vue3';

  import  ModalCustomized from './Modal.vue';

  import MyModal from './MyModal.vue';

  import SearchName from './SearchName.vue';

const form\_search = useForm({

       adder\_name:'',

       adder\_id:null,

       adder\_results:null,

       verifier\_name:'',

       viewer\_id:null,

       viewer\_results:null,

       viewer\_name:'',

       verifier\_id: null,

       verifier\_results: null,

       title:'',

       operation\_id: null

})

const form = useForm({

      title:'',

      type:'',

      adder\_id:null,

      viewer\_id:null,

      verifier\_id: null,

      menu\_id: null,

      min: null,

      max: null

})

  watchEffect(()=>{

    form.menu\_id = props.menu?.id ;

  })

  const showModal = ref(false);

  const showSearchBox = reactive({

      'adder':false,

      'verifier': false,

      'viewer': false

  })

  function close\_seach\_fn(){

        showSearchBox.adder = false

        showSearchBox.verifier = false

        showSearchBox.viewer = false

  }

  const updaterResults = (selected\_result, field\_name)=>{

    if (field\_name=="adder")  {

         form.adder\_id= selected\_result.id

         close\_seach\_fn()

         console.log('updaterResults: ',selected\_result)

    }

    if (field\_name=="verifier") {

        form.verifier\_id= selected\_result.id

        close\_seach\_fn()

    }

    if (field\_name=="viewer") {

       form.viewer\_id= selected\_result.id

       close\_seach\_fn()

    }

          form.errors=""

          axios({

          method: "post",

          url: `/search/select/${selected\_result.id}`,

          data: {

            username: "user",

            password: "pass",

          },

          }).then((response) => {

            console.log('json response:: ',response);

          }).catch(error=>{

                        if (error.response) {

                    // The request was made and the server responded with a status code

                                 if(error.response.status==422){

                                   // validation error

                                   toast.error(error.response.data.message || 'خطا در  اعتبارسنجی اطلاعات ورودی  ')

                                   form.errors=Object.values(error.response.data?.errors).flat()

                                 }else{

                                   toast.error(error.response.data.message || 'خطای نامشخص')

                                 }

                            //.error('Error Status:', error.response.status);

                            console.error('error.response.data', error.response.data,'error.response.status: ',error.response.status);

                           // console.error('Error Message:', error.response.data.message || 'No specific message provided');

                       } else if (error.request) {

                    // The request was made but no response was received

                          console.error('No response received:', error.request);

                          toast.error('خطایی اتفاق افتاده است')

                      } else {

                    // Something happened in setting up the request

                            console.error('Error:', error.message);

                            toast.error('خطایی اتفاق افتاده است')

                     }

          });

    console.log('selected\_result: ',selected\_result)

  }

//showToast('success','موفق شدید ')

watchEffect(() => {

  console.log("Modal visibility:", showModal.value);

});

const sendData = () => {

  //  console.log('form: ',form)

  // return ;

      form.post('/admin/item/save',{

            onError: (errors) => {

                  toast.error('An error occurred. Please try again.');

                  console.log('error: ',errors)

          },

            onSuccess:(response)=>{

              toast.success('عملیات جدید با موفقیت ساخته شد')

              form.clearErrors()

            }

        })

 };

const handleAdderNameChange =  \_.debounce(()=>{

      axios.get(`/search/user/name?q=${form\_search.adder\_name}`).then(({data})=>{

        console.log('api success, data : ',data)

      }).catch(error=>{

         console.log('error in API. error: ',error)

      })

},500)

  </script>

  <style>

  .color\_red {

    color: red;

  }

/\* Add more specific CSS rules to style the toast messages \*/

.vue-toastify {

  position: fixed;

  top: 20px;

  right: 20px;

  z-index: 9999;

}

.vue-toastify .toast {

  margin-bottom: 10px;

  border-radius: 5px;

  padding: 10px 20px;

}

.vue-toastify .toast.toast-success {

  background-color: #4caf50; /\* Green background for success messages \*/

  color: #fff; /\* White text color for success messages \*/

}

.vue-toastify .toast.toast-error {

  background-color: #f44336; /\* Red background for error messages \*/

  color: #fff; /\* White text color for error messages \*/

}

.vue-toastify .toast.toast-warning {

  background-color: #ff9800; /\* Orange background for warning messages \*/

  color: #fff; /\* White text color for warning messages \*/

}

  /\* Add your custom styles here or use Bootstrap classes \*/

  .w\_500 {

    width: 500px;

  }

  </style>