 <div class="add-category-container">

    <h2 class="header">Create New category</h2>

    <div class="separator"></div>

    <form [formGroup]="categoryForm">

      <div class="form-group">

        <label for="name">Name</label>

        <input

          type="text"

          id="name"

          formControlName="name"

          placeholder="Enter Name"

          class="form-control"

          [ngClass]="{'invalid': categoryForm.get('name')?.invalid && categoryForm.get('name')?.touched}">

      </div>

      <div class="form-group">

        <label for="subSet">Sub Set</label>

        <input

          type="text"

          id="subSet"

          formControlName="subSet"

          placeholder="Enter Sub Set"

          class="form-control"

          [ngClass]="{'invalid': categoryForm.get('subSet')?.invalid && categoryForm.get('subSet')?.touched}">

      </div>

      <div class="form-group">

        <label for="image">Image</label>

        <div class="file-input-container">

          <label for="file-upload" class="file-input-label">Choose File</label>

          <input

            type="file"

            id="file-upload"

            (change)="onFileSelected($event)"

            accept="image/\*"

            class="file-input">

          <span class="file-name">{{ selectedFileName }}</span>

        </div>

      </div>

      <div class="actions">

        <button type="button" class="btn cancel-btn" (click)="cancel()">Cancel</button>

        <button type="button" class="btn save-btn" (click)="saveCategory()" [disabled]="categoryForm.invalid">Save Changes</button>

      </div>

    </form>

  </div>

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';

import { CategoriesServiceService } from '../../../services/Categories/categories-service.service';

import { CommonModule } from '@angular/common';

import { Router } from '@angular/router';

@Component({

  selector: 'app-add-category',

  standalone: true,

  imports: [CommonModule, ReactiveFormsModule],

  templateUrl: './add-category.component.html',

  styleUrls: ['./add-category.component.css']

})

export class AddCategoryComponent implements OnInit {

  categoryForm!: FormGroup;

  selectedFile: File | null = null;

  selectedFileName: string = 'No file chosen';

  categories: any[] = [];

  constructor(private fb: FormBuilder, private categoryService: CategoriesServiceService, private router: Router) {}

  ngOnInit(): void {

    this.initForm();

    this.loadCategories();

  }

  initForm(): void {

    this.categoryForm = this.fb.group({

      name: ['', [Validators.required]],

      subSet: ['', [Validators.required]],

      image: [null]

    });

  }

  loadCategories(): void {

    this.categoryService.getCategories().subscribe({

      next: (data) => {

        this.categories = data;

      },

      error: (error) => {

        console.error('Error fetching categories:', error);

      }

    });

  }

  onFileSelected(event: Event): void {

    const input = event.target as HTMLInputElement;

    if (input.files && input.files.length) {

      this.selectedFile = input.files[0];

      this.selectedFileName = this.selectedFile.name;

      this.categoryForm.patchValue({ image: this.selectedFile });

    }

  }

  saveCategory(): void {

    if (this.categoryForm.valid) {

      const formData = new FormData();

      formData.append('name', this.categoryForm.get('name')?.value || '');

      formData.append('subset', this.categoryForm.get('subSet')?.value || '');

      if (this.selectedFile) {

        formData.append('ImageFile', this.selectedFile);

      }

      console.log('FormData Before Sending:', [...formData.entries()]); // Debugging

      this.categoryService.addCategory(formData).subscribe({

        next: (response) => {

          console.log('Category created successfully', response);

          this.loadCategories();

          this.resetForm();

          this.router.navigate(['dashboard/course/viewCategory']); // ✅ Redirect

        },

        error: (error) => {

          console.error('Error creating category:', error);

          alert('Failed to create category. Please try again.');

        }

      });

    } else {

      this.markFormGroupTouched(this.categoryForm);

    }

  }

  cancel(): void {

    this.resetForm();

  }

  resetForm(): void {

    this.categoryForm.reset();

    this.selectedFile = null;

    this.selectedFileName = 'No file chosen';

  }

  markFormGroupTouched(formGroup: FormGroup): void {

    Object.values(formGroup.controls).forEach(control => {

      control.markAsTouched();

      if ((control as any).controls) {

        this.markFormGroupTouched(control as FormGroup);

      }

    });

  }

}

.add-category-container {

    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

    max-width: 100%;

    padding: 20px;

    background-color: white;

    border-radius: 8px;

    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);

  }

  .header {

    font-size: 20px;

    font-weight: 600;

    color: #333;

    margin: 0;

    padding: 0 0 10px 0;

  }

  .separator {

    height: 1px;

    background-color: #e0e0e0;

    margin: 10px 0 20px 0;

  }

  .form-group {

    margin-bottom: 20px;

  }

  label {

    display: block;

    margin-bottom: 8px;

    font-weight: 500;

    color: #333;

    font-size: 14px;

  }

  .form-control {

    width: 100%;

    padding: 10px 12px;

    border: 1px solid #d1d1d1;

    border-radius: 4px;

    font-size: 14px;

    box-sizing: border-box;

  }

  .form-control:focus {

    border-color: #7b1fa2;

    outline: none;

  }

  .form-control.invalid {

    border-color: #f44336;

  }

  .file-input-container {

    display: flex;

    align-items: center;

    border: 1px solid #d1d1d1;

    border-radius: 4px;

    overflow: hidden;

    background-color: #fff;

  }

  .file-input-label {

    display: inline-block;

    padding: 10px 12px;

    background-color: #f5f5f5;

    border-right: 1px solid #d1d1d1;

    cursor: pointer;

    margin: 0;

    font-weight: normal;

    white-space: nowrap;

  }

  .file-input {

    position: absolute;

    width: 1px;

    height: 1px;

    padding: 0;

    margin: -1px;

    overflow: hidden;

    clip: rect(0, 0, 0, 0);

    border: 0;

  }

  .file-name {

    padding: 0 12px;

    color: #666;

    flex-grow: 1;

    white-space: nowrap;

    overflow: hidden;

    text-overflow: ellipsis;

    font-size: 14px;

  }

  .actions {

    display: flex;

    justify-content: flex-end;

    margin-top: 30px;

    gap: 10px;

  }

  .btn {

    padding: 10px 20px;

    border: none;

    border-radius: 20px;

    font-size: 14px;

    font-weight: 500;

    cursor: pointer;

    transition: background-color 0.2s;

  }

  .cancel-btn {

    background-color: transparent;

    color: #7b1fa2;

    border: 1px solid #7b1fa2;

  }

  .save-btn {

    background-color: #7b1fa2;

    color: white;

  }

  .cancel-btn:hover {

    background-color: rgba(123, 31, 162, 0.1);

  }

  .save-btn:hover {

    background-color: #6a1b9a;

  }

  .save-btn:disabled {

    background-color: #cccccc;

    color: #666666;

    cursor: not-allowed;

  }

  @media (max-width: 768px) {

    .actions {

      flex-direction: column;

    }

    .btn {

      width: 100%;

      margin-bottom: 10px;

    }

  }

Categories -> categoriesService

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class CategoriesServiceService {

  private baseUrl = 'https://localhost:7264/api/Categories'; // ✅ Correct API URL

  constructor(private http: HttpClient) {}

  // Fetch all categories

  getCategories(): Observable<any[]> {

    return this.http.get<any[]>(`https://localhost:7264/api/Categories`);

  }

  // Get a specific category by ID

  getCategoryById(id: number): Observable<any> {

    return this.http.get<any>(`${this.baseUrl}/${id}`);

  }

  // Add a new category with image upload

  addCategory(formData: FormData){

    return this.http.post(`${this.baseUrl}`, formData);

  }

  // Update an existing category

  updateCategory(id: number, formData: FormData): Observable<any> {

    return this.http.put<any>(`${this.baseUrl}/${id}`, formData);

  }

}

View category

<div class="view-category-container">

    <div class="search-container">

      <label for="search">Search:</label>

      <input

        type="text"

        id="search"

        [(ngModel)]="searchTerm"

        (input)="search()"

        placeholder="Search categories...">

    </div>

    <table class="categories-table">

      <thead>

        <tr>

          <th class="image-column">Image</th>

          <th>Name</th>

          <th>Sub Set</th>

          <th>Status</th>

          <th>Actions</th>

        </tr>

      </thead>

      <tbody>

        <tr \*ngFor="let category of filteredCategories">

          <td class="image-column">

            <div class="category-image-container">

              <!-- <img [src]="category.imageUrl" alt="{{ category.name }}" class="category-image"> -->

              <img [src]="category.imagepath" alt="{{ category.name }}" class="category-image">

            </div>

          </td>

          <td>{{ category.name }}</td>

          <td>{{ category.subset }}</td>

          <td>

            <div class="status-badge" [ngClass]="{'active': category.status === 'Active', 'inactive': category.status === 'Inactive'}">

              <span class="status-dot"></span>

              <span>{{ category.status ? 'Active' : 'Inactive' }}</span>

            </div>

          </td>

          <td class="actions-column">

            <button class="action-btn edit-btn" (click)="editCategory(category)">Edit</button>

          </td>

        </tr>

      </tbody>

    </table>

    <div class="pagination-container">

      <button

        class="pagination-btn"

        [disabled]="currentPage === 1"

        (click)="goToFirstPage()"

        [ngClass]="{'disabled': currentPage === 1}">

        «

      </button>

      <button

        class="pagination-btn"

        [disabled]="currentPage === 1"

        (click)="goToPreviousPage()"

        [ngClass]="{'disabled': currentPage === 1}">

        ‹

      </button>

      <div class="pagination-pages">

        <button

          \*ngFor="let page of pages"

          class="pagination-btn page-number"

          [ngClass]="{'active': page === currentPage}"

          (click)="goToPage(page)">

          {{ page }}

        </button>

      </div>

      <button

        class="pagination-btn"

        [disabled]="currentPage === totalPages"

        (click)="goToNextPage()"

        [ngClass]="{'disabled': currentPage === totalPages}">

        ›

      </button>

      <button

        class="pagination-btn"

        [disabled]="currentPage === totalPages"

        (click)="goToLastPage()"

        [ngClass]="{'disabled': currentPage === totalPages}">

        »

      </button>

    </div>

  </div>

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import { HttpClient } from '@angular/common/http';

interface Category {

  id: number;

  name: string;

  subset: string;

  imagepath: string;

  status: 'Active' | 'Inactive';

}

@Component({

  selector: 'app-view-category',

  standalone: true,

  imports: [CommonModule, FormsModule],

  templateUrl: './view-category.component.html',

  styleUrls: ['./view-category.component.css']

})

export class ViewCategoryComponent implements OnInit {

  categories: Category[] = [];

  filteredCategories: Category[] = [];

  searchTerm: string = '';

  // Pagination properties

  currentPage: number = 1;

  itemsPerPage: number = 5;

  totalPages: number = 1;

  pages: number[] = [1];

  constructor(private http: HttpClient) {}

ngOnInit(): void {

  this.loadCategories();

}

// loadCategories(): void {

//   this.http.get<{ categories: Category[] }>('https://localhost:7264/api/Categories').subscribe({

//   next: (data) => {

//     this.categories = data.categories; // ✅ Extract nested array

//     this.filterCategories();

//     this.calculateTotalPages();

//   },

//   error: (error) => {

//     console.error('Error fetching categories:', error);

//   }

// });

// }

loadCategories(): void {

  this.http.get<{ success: boolean; message: string; data: Category[] }>('https://localhost:7264/api/Categories')

    .subscribe({

      next: (response) => {

        console.log('API Response:', response);

        if (response && response.success && Array.isArray(response.data)) {

          this.categories = response.data; // ✅ Extract data correctly

        } else {

          console.error('Invalid response format:', response);

          this.categories = []; // Set empty array to prevent errors

        }

        this.filterCategories();

        this.calculateTotalPages();

      },

      error: (error) => {

        console.error('Error fetching categories:', error);

      }

    });

}

  search(): void {

    this.currentPage = 1;

    this.filterCategories();

    this.calculateTotalPages();

  }

  filterCategories(): void {

    if (!this.searchTerm.trim()) {

      this.filteredCategories = [...this.categories];

    } else {

      const term = this.searchTerm.toLowerCase().trim();

      this.filteredCategories = this.categories.filter(category =>

        category.name.toLowerCase().includes(term) ||

        category.subset.toLowerCase().includes(term)

      );

    }

    // Apply pagination

    const startIndex = (this.currentPage - 1) \* this.itemsPerPage;

    this.filteredCategories = this.filteredCategories.slice(startIndex, startIndex + this.itemsPerPage);

  }

  calculateTotalPages(): void {

    let filteredCount = 0;

    if (!this.searchTerm.trim()) {

      filteredCount = this.categories.length;

    } else {

      const term = this.searchTerm.toLowerCase().trim();

      filteredCount = this.categories.filter(category =>

        category.name.toLowerCase().includes(term) ||

        category.subset.toLowerCase().includes(term)

      ).length;

    }

    this.totalPages = Math.ceil(filteredCount / this.itemsPerPage);

    this.generatePageArray();

  }

  generatePageArray(): void {

    this.pages = [];

    for (let i = 1; i <= this.totalPages; i++) {

      this.pages.push(i);

    }

  }

  goToPage(page: number): void {

    if (page >= 1 && page <= this.totalPages) {

      this.currentPage = page;

      this.filterCategories();

    }

  }

  goToFirstPage(): void {

    this.goToPage(1);

  }

  goToLastPage(): void {

    this.goToPage(this.totalPages);

  }

  goToPreviousPage(): void {

    this.goToPage(this.currentPage - 1);

  }

  goToNextPage(): void {

    this.goToPage(this.currentPage + 1);

  }

  editCategory(category: Category): void {

    // Implementation for edit functionality

    console.log('Edit category:', category);

  }

}