```typescript
import { Component, Input, OnInit } from '@angular/core';
import { Project } from '../../Models/project.model';
import { Folder } from '../../Models/folder.model';
import { ProjectsService } from '../../projects.service';

@Component({
  selector: 'app-projects-recursion',
  templateUrl: './projects-recursion.component.html',
  styleUrls: ['./projects-recursion.component.scss']
})
export class ProjectsRecursionComponent implements OnInit {
  @Input() project: Project;
  @Input() recursionIndex: number;
  @Input() folder: Folder;
  folderSelection: Folder[];
  itemRecursionLength: number;
  public isCollapsed = [];
  public fileIterations: number[];

  constructor(private projectService: ProjectsService) { }

  // path length brought on par with the recursion standard (2 levels down the tree)
  // create one isCollapsed for each horizontalFolder
  ngOnInit() {
    this.itemRecursionLength = this.folder.pathLength - 3;
    this.fileIterations = Array.from(Array(this.folder.files.length).keys());
    this.folderSelection = this._folderSelector();
    if (this.recursionIndex < this.itemRecursionLength) {
      for (const folder of this.project.folders[this.recursionIndex]) {
        this.isCollapsed.push(true);
      }
      if (this.recursionIndex === 0) {
        this.isCollapsed[0] = false;
      }
    }
  }

  // select folders by checking that the part ahead of the index is equal to the
      current folder path
  private _folderSelector(): Folder[] {
    const folderSelection: Folder[] = [];

    if (this.project.folders[this.recursionIndex]) {
      for (const loopFolder of this.project.folders[this.recursionIndex]) {
        if (this.projectService.isPathwayEqualUpToIndex(
            this.projectService.getFilelessPath(loopFolder.folderPath, true),
            this.projectService.getFilelessPath(this.folder.folderPath, true),
            this.recursionIndex)) {

          folderSelection.push(loopFolder);
        }
      }
    }
    return folderSelection;
  }

  // return file extention for a given path
  getExtension(path: string) {
    return path.substring(path.length - 3);
  }

  // return Font Awesome icon classes based on the type of the file
  getFaIcon (path: string) {
    const extension = this.getExtension(path);

    if (extension === 'jpg' || extension === 'png') {
      return 'fa fa-file-image-o charcoal';
    } else if (extension === 'pdf') {
      return 'fa fa-file-pdf-o red';
    }
  }
}

```