Dynamic nested menu system in Angular v11 using Angular Material's mat-menu with matMenuTriggerFor, where a menu can contain children that recursively open sub-menus.

This is a common pattern for hierarchical menus (like categories, folder navigation, etc.).
Let me walk you through a clean recursive approach:

---

1. Example Menu Data Model

Define a recursive data structure in your parent component:

```
// menu.model.ts
export interface MenuItem {
  label: string;
  icon?: string;
  action?: () => void;   // optional action when clicked
  children?: MenuItem[]; // recursive children
}
```

---

2. Sample Menu Data

In your parent component:

```
// top-menu.component.ts
import { Component } from '@angular/core';
import { MenuItem } from './menu.model';

@Component({
  selector: 'app-top-menu',
  templateUrl: './top-menu.component.html',
})
export class TopMenuComponent {
  menuItems: MenuItem[] = [
    {
      label: 'File',
      children: [
        { label: 'New', action: () => alert('New clicked') },
        { label: 'Open', action: () => alert('Open clicked') },
        {
```

```
        label: 'Recent',
        children: [
          { label: 'File1.txt', action: () => alert('File1 clicked') },
          { label: 'File2.txt', action: () => alert('File2 clicked') },
        ],
      },
    ],
  },
  {
    label: 'Edit',
    children: [
      { label: 'Cut', action: () => alert('Cut clicked') },
      { label: 'Copy', action: () => alert('Copy clicked') },
    ],
  },
];
}
```

---

3. Recursive Child Component

This is the key: the child renders itself recursively.

```
// nested-menu.component.ts
import { Component, Input } from '@angular/core';
import { MenuItem } from '../menu.model';

@Component({
  selector: 'app-nested-menu',
  templateUrl: './nested-menu.component.html',
})
export class NestedMenuComponent {
  @Input() items: MenuItem[];
}
```

---

4. Recursive Child Template

We'll use Angular Material menus and matMenuTriggerFor to recursively bind children.

```
<!-- nested-menu.component.html -->
<ng-template #recursiveMenu let-items>
  <mat-menu>
    <ng-container *ngFor="let item of items">
      <!-- If item has children -->
      <button mat-menu-item *ngIf="item.children" [matMenuTriggerFor]="childMenu">
        {{ item.label }}
      </button>
      <ng-template #childMenu="matMenu">
        <ng-container *ngTemplateOutlet="recursiveMenu; context: {$implicit:
item.children}"></ng-container>
      </ng-template>

      <!-- If item is a leaf -->
      <button mat-menu-item *ngIf="!item.children" (click)="item.action?.()">
        {{ item.label }}
      </button>
    </ng-container>
  </mat-menu>
</ng-template>

<!-- Root Menu -->
<ng-container *ngTemplateOutlet="recursiveMenu; context: {$implicit: items}"></ng-container>
```

---

5. Hook into Parent Template

In your top-menu.component.html, you can now place the recursive menu:

```
<!-- top-menu.component.html -->
<app-nested-menu [items]="menuItems"></app-nested-menu>
```

And trigger it with a button (like in a navbar):

```
<button mat-button [matMenuTriggerFor]="rootMenu">Menu</button>
<app-nested-menu #rootMenu="matMenu" [items]="menuItems"></app-nested-menu>
```

✅ How it works:

The child component (nested-menu) defines a mat-menu.

If a menu item has children, it attaches matMenuTriggerFor and recursively renders another mat-menu.

If it’s a leaf, it simply calls the action.

This way, you can nest arbitrarily deep menus dynamically.

⚡ Tip: If you want icons or custom templates, just expand the <button mat-menu-item> template.

Would you like me to adjust this to work with your existing top menu parent-child recursive component setup (instead of introducing a new recursive component), or do you prefer this standalone recursive menu component approach?