M**ulti-level dropdown menu** with both hover and click functionality, built on top of Angular Material (`mat-toolbar` + `mat-menu`) in Angular v11.

Angular Material menus (`mat-menu`) support nesting, but you'll need a structured approach to handle **dynamic levels** and **hover + click triggers**.

Here's a **working pattern** you can adapt:

---

1. Define Your Menu Model

Create a hierarchical model for menus:

```
// menu.model.ts
export interface MenuItem {
  label: string;
  icon?: string;
  route?: string;
  children?: MenuItem[];
}
```

---

2. Example Menu Data

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

@Component({
  selector: 'app-top-menu',
  templateUrl: './top-menu.component.html',
  styleUrls: ['./top-menu.component.scss']
})
export class TopMenuComponent {
  menuItems: MenuItem[] = [
    {
      label: 'Products',
      children: [
        {
          label: 'Electronics',
          children: [
```

```
          { label: 'Mobiles', route: '/products/mobiles' },
          { label: 'Laptops', route: '/products/laptops' }
        ]
      },
      {
        label: 'Furniture',
        children: [
          { label: 'Tables', route: '/products/tables' },
          { label: 'Chairs', route: '/products/chairs' }
        ]
      }
    ]
  },
  {
    label: 'Services',
    children: [
      { label: 'Consulting', route: '/services/consulting' },
      { label: 'Support', route: '/services/support' }
    ]
  },
  { label: 'About Us', route: '/about' }
 ];
}
```

---

3. Recursive Menu Template

Use recursion with `ng-template` to support infinite nesting:

```
<!-- top-menu.component.html -->
<mat-toolbar color="primary">
 <ng-container *ngFor="let item of menuItems">
  <button mat-button
          [matMenuTriggerFor]="item.children ? menuMap.get(item) : null"
          [routerLink]="!item.children ? item.route : null">
    {{ item.label }}
  </button>

  <!-- Attach dropdown if children exist -->
  <mat-menu #menu="matMenu">
   <ng-template matMenuContent>
    <ng-container *ngFor="let child of item.children">
     <button mat-menu-item
```

```
            [matMenuTriggerFor]="child.children ? menuMap.get(child) : null"
            [routerLink]="!child.children ? child.route : null">
          {{ child.label }}
        </button>
      </ng-container>
    </ng-template>
  </mat-menu>
 </ng-container>
</mat-toolbar>
```

---

4. Hover + Click Support

Angular Material menus are **click-only by default**. To enable **hover**:

```
// top-menu.component.ts (continued)
import { AfterViewInit, QueryList, ViewChildren } from '@angular/core';
import { MatMenuTrigger } from '@angular/material/menu';

export class TopMenuComponent implements AfterViewInit {
  @ViewChildren(MatMenuTrigger) triggers!: QueryList<MatMenuTrigger>;

  ngAfterViewInit() {
    this.triggers.forEach(trigger => {
      const button = trigger._element.nativeElement;

      button.addEventListener('mouseenter', () => {
        if (!trigger.menuOpen) trigger.openMenu();
      });

      trigger.menu?.closed.subscribe(() => {
        // Close logic if you want hover-out to close
      });
    });
  }
}
```

This way:

- **Click** still works (default behavior).
- **Hover** will also open the menu.

---

5. Styling (Optional)

To make hover menus appear smoothly and properly aligned:

```
::ng-deep .mat-menu-panel {
  min-width: 200px;
}

::ng-deep .mat-menu-item {
  font-size: 14px;
}

::ng-deep .mat-toolbar button {
  color: white;
  font-weight: 500;
}
```

✅ With this setup:

- `mat-toolbar` shows **top-level menus**.
- Hover or click on any menu item to open its dropdown.
- Submenus can go infinitely deep (recursive).
- Supports **routing** (`routerLink`) for leaf items.

Do you want me to build this so the **child menus are rendered dynamically via a recursive component** (cleaner approach for large hierarchies), instead of nesting templates inline?