I understand your challenge. You need to migrate a recursive menu component from Angular 7 to Angular 11 where the API for matMenuTriggerFor has changed to require a MatMenu instance rather than a component.

Here's how you can refactor your recursive menu component to work with Angular 11:

1. Update Your Child Component Structure

```typescript
// child-menu.component.ts
import { Component, Input, Output, EventEmitter, ViewChild } from '@angular/core';
import { MatMenu } from '@angular/material/menu';

@Component({
  selector: 'app-child-menu',
  template: `
    <mat-menu #menu="matMenu" class="nested-menu">
      <ng-template matMenuContent>
        <ng-container *ngFor="let item of menuItems">
          <button *ngIf="!item.children || item.children.length === 0"
                  mat-menu-item
                  (click)="handleClick(item)">
            {{ item.label }}
          </button>

          <app-child-menu *ngIf="item.children && item.children.length > 0"
                          [menuItems]="item.children"
                          [parentMenu]="menu"
                          (itemSelected)="handleClick($event)">
          </app-child-menu>
        </ng-container>
      </ng-template>
    </mat-menu>
  `
})
export class ChildMenuComponent {
  @Input() menuItems: any[];
  @Input() parentMenu: MatMenu;
  @Output() itemSelected = new EventEmitter<any>();

  @ViewChild('menu', { static: true }) menu: MatMenu;

  handleClick(item: any) {
    this.itemSelected.emit(item);
```

```
  }
}
```

2. Update Your Parent Component

```typescript
// parent.component.ts
import { Component, ViewChild } from '@angular/core';
import { MatMenu } from '@angular/material/menu';

@Component({
  selector: 'app-parent',
  template: `
    <mat-toolbar>
      <ng-container *ngFor="let item of topLevelMenuItems">
        <button *ngIf="!item.children || item.children.length === 0"
                mat-button
                (click)="handleClick(item)">
          {{ item.label }}
        </button>

        <button *ngIf="item.children && item.children.length > 0"
                mat-button
                [matMenuTriggerFor]="getChildMenu(item)">
          {{ item.label }}
        </button>
      </ng-container>
    </mat-toolbar>

    <!-- Hidden child components for each menu -->
    <div style="display: none;">
      <app-child-menu *ngFor="let item of topLevelMenuItemsWithChildren"
                      [menuItems]="item.children"
                      #childMenu>
      </app-child-menu>
    </div>
  `
})
export class ParentComponent {
  topLevelMenuItems: any[] = [/* your menu data */];

  @ViewChild('childMenu', { static: false }) childMenu: any;
```

```
  get topLevelMenuItemsWithChildren() {
    return this.topLevelMenuItems.filter(item =>
      item.children && item.children.length > 0
    );
  }

  getChildMenu(menuItem: any): MatMenu {
    // You'll need to implement logic to find the correct child menu instance
    // This might require using @ViewChildren and tracking menu items
    return this.childMenu?.menu;
  }

  handleClick(item: any) {
    // Handle menu item selection
  }
}
```

3. Alternative Approach Using Template References

```typescript
// parent.component.ts (alternative approach)
import { Component, ViewChildren, QueryList } from '@angular/core';
import { ChildMenuComponent } from './child-menu.component';

@Component({
  selector: 'app-parent',
  template: `
    <mat-toolbar>
      <ng-container *ngFor="let item of topLevelMenuItems; let i = index">
        <button *ngIf="!item.children || item.children.length === 0"
                mat-button
                (click)="handleClick(item)">
          {{ item.label }}
        </button>

        <button *ngIf="item.children && item.children.length > 0"
                mat-button
                [matMenuTriggerFor]="childMenus.toArray()[i]?.menu">
          {{ item.label }}
        </button>
      </ng-container>
    </mat-toolbar>
```

```
    <div style="display: none;">
      <app-child-menu *ngFor="let item of topLevelMenuItems"
                      [menuItems]="item.children || []"
                      #childMenu>
      </app-child-menu>
    </div>
  `
})
export class ParentComponent {
  topLevelMenuItems: any[] = [/* your menu data */];

  @ViewChildren('childMenu') childMenus: QueryList<ChildMenuComponent>;

  handleClick(item: any) {
    // Handle menu item selection
  }
}
```

4. Enhanced Child Component with Better Recursion Handling

```typescript
// child-menu.component.ts (enhanced)
import { Component, Input, Output, EventEmitter, ViewChild } from '@angular/core';
import { MatMenu } from '@angular/material/menu';

@Component({
  selector: 'app-child-menu',
  template: `
    <mat-menu #menu="matMenu" class="nested-menu">
      <ng-template matMenuContent>
        <ng-container *ngFor="let item of menuItems">
          <button *ngIf="!item.children || item.children.length === 0"
                  mat-menu-item
                  (click)="handleClick(item)">
            {{ item.label }}
          </button>

          <button *ngIf="item.children && item.children.length > 0"
                  mat-menu-item
                  [matMenuTriggerFor]="childMenu?.menu">
            {{ item.label }}
          </button>
```

```
        <app-child-menu *ngIf="item.children && item.children.length > 0"
                        [menuItems]="item.children"
                        #childMenu
                        style="display: none;">
        </app-child-menu>
       </ng-container>
     </ng-template>
   </mat-menu>
  `
})
export class ChildMenuComponent {
  @Input() menuItems: any[];
  @Output() itemSelected = new EventEmitter<any>();

  @ViewChild('menu', { static: true }) menu: MatMenu;
  @ViewChild('childMenu', { static: false }) childMenu: ChildMenuComponent;

  handleClick(item: any) {
    this.itemSelected.emit(item);
  }
}
```

5. Service-Based Approach (Recommended)

Create a service to manage menu instances:

```typescript
// menu.service.ts
import { Injectable } from '@angular/core';
import { MatMenu } from '@angular/material/menu';

@Injectable({
  providedIn: 'root'
})
export class MenuService {
  private menuInstances = new Map<string, MatMenu>();

  registerMenu(key: string, menu: MatMenu) {
    this.menuInstances.set(key, menu);
  }

  getMenu(key: string): MatMenu | undefined {
    return this.menuInstances.get(key);
```

```
  }

  unregisterMenu(key: string) {
    this.menuInstances.delete(key);
  }
}
```

Key Migration Steps:

1. Update Angular Material: Make sure you've properly updated Angular Material to v11
2. Replace Component References: Change matMenuTriggerFor="componentName" to [matMenuTriggerFor]="menuInstance"
3. Use ViewChild/ViewChildren: Access the MatMenu instance from your child component
4. Handle Recursion: Ensure nested components properly reference their own menu instances
5. Update Templates: Modify your templates to work with the new API

The main concept is that instead of passing the component itself to matMenuTriggerFor, you now need to pass the MatMenu instance that the component contains.

Choose the approach that best fits your existing code structure. The service-based approach is often the most maintainable for complex recursive menus.

I'll create a zip file containing all the necessary files for your Angular 11 menu migration. Here's the complete file structure:

File Structure

```
angular-menu-migration/
├── src/
│   ├── app/
│   │   ├── parent/
│   │   │   ├── parent.component.ts
│   │   │   ├── parent.component.html
│   │   │   ├── parent.component.css
│   │   │   └── parent.component.spec.ts
│   │   ├── child-menu/
│   │   │   ├── child-menu.component.ts
│   │   │   ├── child-menu.component.html
│   │   │   ├── child-menu.component.css
│   │   │   └── child-menu.component.spec.ts
│   │   ├── services/
│   │   │   ├── menu.service.ts
│   │   │   └── menu.service.spec.ts
│   │   ├── models/
│   │   │   └── menu-item.model.ts
│   │   └── app.module.ts
│   └── assets/
│       └── menu-data.json
├── package.json
├── angular.json
└── README.md
```

Files Content

1. package.json

```json
{
  "name": "angular-menu-migration",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
```

```
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "dependencies": {
    "@angular/animations": "^11.0.0",
    "@angular/cdk": "^11.0.0",
    "@angular/common": "^11.0.0",
    "@angular/compiler": "^11.0.0",
    "@angular/core": "^11.0.0",
    "@angular/forms": "^11.0.0",
    "@angular/material": "^11.0.0",
    "@angular/platform-browser": "^11.0.0",
    "@angular/platform-browser-dynamic": "^11.0.0",
    "@angular/router": "^11.0.0",
    "rxjs": "~6.6.0",
    "tslib": "^2.0.0",
    "zone.js": "~0.10.2"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.1100.0",
    "@angular/cli": "~11.0.0",
    "@angular/compiler-cli": "^11.0.0",
    "@types/jasmine": "~3.6.0",
    "@types/node": "^12.11.1",
    "codelyzer": "^6.0.0",
    "jasmine-core": "~3.6.0",
    "jasmine-spec-reporter": "~5.0.0",
    "karma": "~5.0.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.0.3",
    "karma-jasmine": "~4.0.0",
    "karma-jasmine-html-reporter": "^1.5.0",
    "protractor": "~7.0.0",
    "ts-node": "~8.3.0",
    "tslint": "~6.1.0",
    "typescript": "~4.0.2"
  }
}
```

2. menu-item.model.ts

```typescript
```

```
export interface MenuItem {
  id: string;
  label: string;
  route?: string;
  icon?: string;
  children?: MenuItem[];
  data?: any;
}
```

3. menu.service.ts

```typescript
import { Injectable } from '@angular/core';
import { MatMenu } from '@angular/material/menu';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class MenuService {
  private menuInstances = new Map<string, MatMenu>();
  private menuItemSelected = new BehaviorSubject<any>(null);

  menuItemSelected$ = this.menuItemSelected.asObservable();

  registerMenu(key: string, menu: MatMenu): void {
    this.menuInstances.set(key, menu);
  }

  getMenu(key: string): MatMenu | undefined {
    return this.menuInstances.get(key);
  }

  unregisterMenu(key: string): void {
    this.menuInstances.delete(key);
  }

  notifyItemSelected(item: any): void {
    this.menuItemSelected.next(item);
  }
}
```

4. child-menu.component.ts

```typescript
import { Component, Input, Output, EventEmitter, ViewChild, OnInit } from '@angular/core';
import { MatMenu } from '@angular/material/menu';
import { MenuItem } from '../../models/menu-item.model';
import { MenuService } from '../../services/menu.service';

@Component({
  selector: 'app-child-menu',
  templateUrl: './child-menu.component.html',
  styleUrls: ['./child-menu.component.css']
})
export class ChildMenuComponent implements OnInit {
  @Input() menuItems: MenuItem[] = [];
  @Input() menuId!: string;
  @Output() itemSelected = new EventEmitter<MenuItem>();

  @ViewChild('menu', { static: true }) menu!: MatMenu;

  constructor(private menuService: MenuService) {}

  ngOnInit(): void {
    if (this.menuId) {
      this.menuService.registerMenu(this.menuId, this.menu);
    }
  }

  handleClick(item: MenuItem): void {
    this.itemSelected.emit(item);
    this.menuService.notifyItemSelected(item);
  }

  getChildMenuId(item: MenuItem): string {
    return `submenu-${item.id}`;
  }
}
```

5. child-menu.component.html

```html
<mat-menu #menu="matMenu" class="nested-menu">
  <ng-template matMenuContent>
```

```
    <ng-container *ngFor="let item of menuItems">
      <!-- Leaf node -->
      <button *ngIf="!item.children || item.children.length === 0"
              mat-menu-item
              (click)="handleClick(item)">
        <mat-icon *ngIf="item.icon">{{ item.icon }}</mat-icon>
        <span>{{ item.label }}</span>
      </button>

      <!-- Node with children -->
      <button *ngIf="item.children && item.children.length > 0"
              mat-menu-item
              [matMenuTriggerFor]="menuService.getMenu(getChildMenuId(item))">
        <mat-icon *ngIf="item.icon">{{ item.icon }}</mat-icon>
        <span>{{ item.label }}</span>
        <mat-icon>arrow_right</mat-icon>
      </button>
    </ng-container>
  </ng-template>
</mat-menu>

<!-- Hidden child menus -->
<div style="display: none;">
  <app-child-menu *ngFor="let item of menuItems"
                  [menuItems]="item.children || []"
                  [menuId]="getChildMenuId(item)"
                  (itemSelected)="handleClick($event)">
  </app-child-menu>
</div>
```

6. parent.component.ts

```typescript
import { Component, OnInit, ViewChildren, QueryList } from '@angular/core';
import { ChildMenuComponent } from '../child-menu/child-menu.component';
import { MenuItem } from '../../models/menu-item.model';
import { MenuService } from '../../services/menu.service';

@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
```

```
export class ParentComponent implements OnInit {
  topLevelMenuItems: MenuItem[] = [];

  @ViewChildren(ChildMenuComponent) childMenus!: QueryList<ChildMenuComponent>;

  constructor(private menuService: MenuService) {}

  ngOnInit(): void {
    this.loadMenuData();
    this.menuService.menuItemSelected$.subscribe(item => {
      this.handleMenuItemSelection(item);
    });
  }

  loadMenuData(): void {
    // Example data - replace with your actual data
    this.topLevelMenuItems = [
      {
        id: '1',
        label: 'Home',
        route: '/home',
        icon: 'home'
      },
      {
        id: '2',
        label: 'Products',
        icon: 'shopping_cart',
        children: [
          {
            id: '2-1',
            label: 'Electronics',
            children: [
              { id: '2-1-1', label: 'Phones', route: '/products/phones' },
              { id: '2-1-2', label: 'Laptops', route: '/products/laptops' }
            ]
          },
          {
            id: '2-2',
            label: 'Clothing',
            children: [
              { id: '2-2-1', label: 'Men', route: '/products/men' },
              { id: '2-2-2', label: 'Women', route: '/products/women' }
            ]
          }
```

```
        ]
      },
      {
        id: '3',
        label: 'Services',
        icon: 'build',
        children: [
          { id: '3-1', label: 'Consulting', route: '/services/consulting' },
          { id: '3-2', label: 'Support', route: '/services/support' }
        ]
      }
    ];
  }

  getMenuId(item: MenuItem): string {
    return `menu-${item.id}`;
  }

  handleClick(item: MenuItem): void {
    this.menuService.notifyItemSelected(item);
  }

  handleMenuItemSelection(item: MenuItem): void {
    console.log('Menu item selected:', item);
    // Handle navigation or other actions
    if (item.route) {
      // this.router.navigate([item.route]);
    }
  }

  getTopLevelMenuItemsWithChildren(): MenuItem[] {
    return this.topLevelMenuItems.filter(item =>
      item.children && item.children.length > 0
    );
  }
}
```

7. parent.component.html

```html
<mat-toolbar color="primary">
  <span>My Application</span>
```

```
  <nav class="toolbar-nav">
    <ng-container *ngFor="let item of topLevelMenuItems">
      <!-- Leaf node -->
      <button *ngIf="!item.children || item.children.length === 0"
              mat-button
              (click)="handleClick(item)"
              [routerLink]="item.route">
        <mat-icon *ngIf="item.icon">{{ item.icon }}</mat-icon>
        {{ item.label }}
      </button>

      <!-- Node with children -->
      <button *ngIf="item.children && item.children.length > 0"
              mat-button
              [matMenuTriggerFor]="menuService.getMenu(getMenuId(item))">
        <mat-icon *ngIf="item.icon">{{ item.icon }}</mat-icon>
        {{ item.label }}
        <mat-icon>arrow_drop_down</mat-icon>
      </button>
    </ng-container>
  </nav>
</mat-toolbar>

<main>
  <router-outlet></router-outlet>
</main>

<!-- Hidden child menu components -->
<div style="display: none;">
  <app-child-menu *ngFor="let item of topLevelMenuItems"
                  [menuItems]="item.children || []"
                  [menuId]="getMenuId(item)">
  </app-child-menu>
</div>
```

8. parent.component.css

```css
.toolbar-nav {
  margin-left: auto;
  display: flex;
  gap: 8px;
}
```

```
.toolbar-nav button {
  display: flex;
  align-items: center;
  gap: 4px;
}
```

9. app.module.ts

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { RouterModule } from '@angular/router';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatButtonModule } from '@angular/material/button';
import { MatMenuModule } from '@angular/material/menu';
import { MatIconModule } from '@angular/material/icon';

import { AppComponent } from './app.component';
import { ParentComponent } from './parent/parent.component';
import { ChildMenuComponent } from './child-menu/child-menu.component';

@NgModule({
  declarations: [
    AppComponent,
    ParentComponent,
    ChildMenuComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    RouterModule.forRoot([]),
    MatToolbarModule,
    MatButtonModule,
    MatMenuModule,
    MatIconModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

10. README.md

```markdown
# Angular Menu Migration to v11

This project demonstrates how to migrate recursive menu components from Angular 7 to Angular 11, where `matMenuTriggerFor` now expects a `MatMenu` instance instead of a component reference.

## Key Changes

1. **Service Pattern**: Uses `MenuService` to manage menu instances
2. **ViewChild References**: Accesses `MatMenu` instances via `@ViewChild`
3. **Recursive Structure**: Maintains the recursive menu functionality
4. **Hidden Containers**: Uses hidden divs to contain recursive components

## Installation

1. Install dependencies: `npm install`
2. Start development server: `ng serve`
3. Navigate to `http://localhost:4200`

## Usage

The menu structure is defined in `parent.component.ts`. Modify the `topLevelMenuItems` array to match your menu structure.

## Migration Notes

- Replace `matMenuTriggerFor="componentName"` with `[matMenuTriggerFor]="service.getMenu(id)"`
- Use `@ViewChild` to access `MatMenu` instances
- Register menus with the service using unique identifiers
```