State Management

State management in angular refers to handling the data (state) shared between components and services in a way that’s predictable, testable and maintainable

Step:1 Create Angular App

* ng new State management

Step:2 Generate Component

* ng g c todo-list
* ng g c add-todo
* ng g s services/todo

todo-list.component.ts file

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

import { Observable } from 'rxjs';

import { Todo, TodoService } from '../services/todo.service';

@Component({

  selector: 'app-todo-list',

  templateUrl: './todo-list.component.html',

  styleUrls: ['./todo-list.component.css']

})

export class TodoListComponent implements OnInit {

  todos$!:Observable<Todo[]>;

  constructor(private todoService:TodoService){}

  ngOnInit(): void {

   this.todos$= this.todoService.todos$;

  }

  removeTask(id:number){

    this.todoService.removeTodo(id);

  }

}

Todo-list.component.html

<div class="container">

    <ul>

        <li \*ngFor="let t of todos$ | async">

            {{t.task}}

            <button class="btn btn-danger" (click)="removeTask(t.id)">Delete</button>

        </li>

    </ul>

</div>

Create todo.service.ts file

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

import { BehaviorSubject } from 'rxjs';

export interface Todo{

  id:number;

  task:string;

}

@Injectable({

  providedIn: 'root'

})

export class TodoService {

  private todos: Todo[]=[];

  private todoSubject= new BehaviorSubject<Todo[]>([]);

  public todos$= this.todoSubject.asObservable()

  constructor() { }

  addTodo(task:string){

    const newTodo:Todo={

      id:Date.now(),

      task,

    };

    this.todos.push(newTodo);

    this.todoSubject.next(this.todos);

  }

  removeTodo(id:number){

    this.todos=this.todos.filter((todo)=>todo.id != id);

    this.todoSubject.next(this.todos);

  }

}

Todo-list.component.ts

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

import { Observable } from 'rxjs';

import { Todo, TodoService } from '../services/todo.service';

@Component({

  selector: 'app-todo-list',

  templateUrl: './todo-list.component.html',

  styleUrls: ['./todo-list.component.css']

})

export class TodoListComponent implements OnInit {

  todos$!:Observable<Todo[]>;

  constructor(private todoService:TodoService){}

  ngOnInit(): void {

   this.todos$= this.todoService.todos$;

  }

  removeTask(id:number){

    this.todoService.removeTodo(id);

  }

}

Todo-list.component.html

<div class="container">

    <ul>

        <li \*ngFor="let t of todos$ | async">

            {{t.task}}

            <button class="btn btn-danger" (click)="removeTask(t.id)">Delete</button>

        </li>

    </ul>

</div>

Add it in app component 🡪 App.component.html

<nav class="navbar navbar-expand-lg bg-body-tertiary">

  <div class="container-fluid">

    <a class="navbar-brand" href="#">Navbar</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

        <li class="nav-item">

          <a class="nav-link active" aria-current="page" href="#">Home</a>

        </li>

        <li class="nav-item">

          <a class="nav-link" routerLink="todo" href="#">Add Todo</a>

        </li>

        <li class="nav-item">

          <a class="nav-link" routerLink="list" href="#">todoList</a>

        </li>

      </ul>

      <form class="d-flex" role="search">

        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search"/>

        <button class="btn btn-outline-success" type="submit">Search</button>

      </form>

    </div>

  </div>

</nav>

<router-outlet></router-outlet>

App.routing.module.ts

const routes: Routes = [

  {path:"list",component:TodoListComponent},

  {path:"todo",component:AddTodoComponent}

];

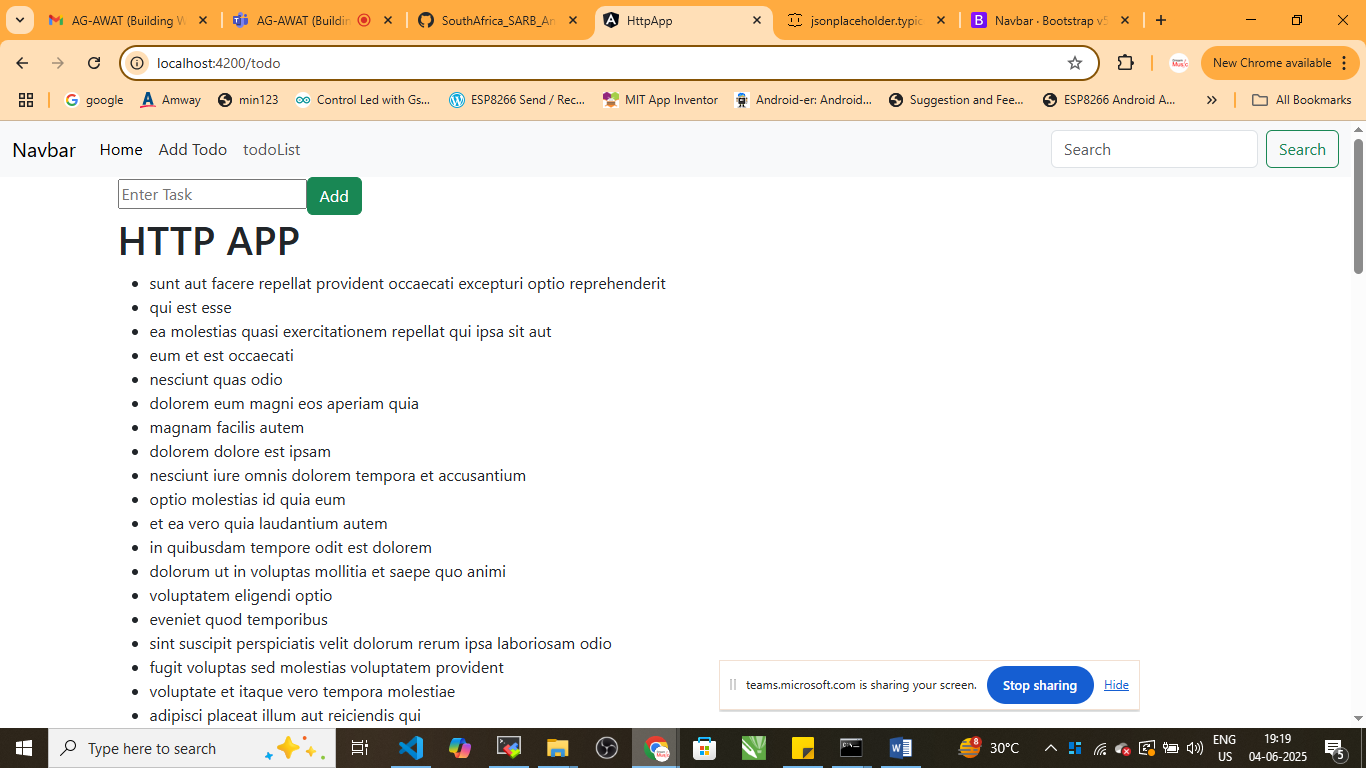
App.module.ts file

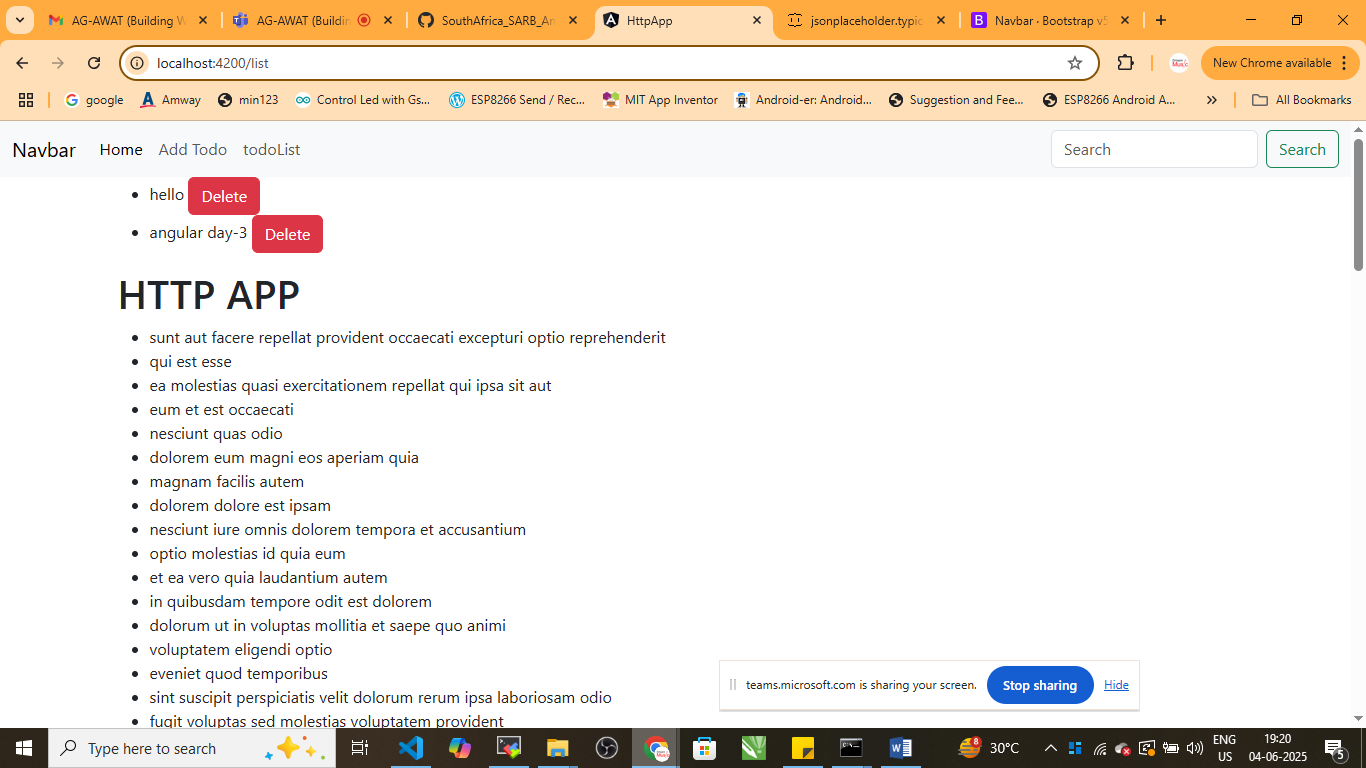
 imports: [

    FormsModule

  ],

Output





Delete the todo

