**Developer Oriented documentation**

Using Angular 17 project that fetches different movie genres from the [TMDB (The Movie Database) API](https://www.themoviedb.org/). The documentation is split into:

**1.] Project Overview**

This Angular 17 application fetches movies from the TMDB API categorized as:

* **Feel Good**
* **Action Fix**
* **Mind Bender**

The application uses a dedicated service to fetch data via HTTP calls and separates display logic into reusable components.

**1.] Folder Structure**

css

CopyEdit

src/

├── app/

│ ├── services/

│ │ └── auth.service.ts

│ ├── components/

├── Index/

│ │ │ └── index.component.ts

│ │ ├── feel-good/

│ │ │ └── feel-good.component.ts

│ │ ├── action-fix/

│ │ │ └── action-fix.component.ts

│ │ ├── details-pages/

│ │ │ └── details-pages.component.ts

├── mind-bender /

│ │ │ └── mind-bender.component.ts

│ └── app.module.ts

├── environments/

│ ├── environment.ts

│ └── environment.prod.ts

**3.] API Integration(TMDB)**

**TMBD Setup**

* Go to <https://www.themoviedb.org/>
* Create an account → Get an **API Key (v3)** from your profile.
* Base URL: https://api.themoviedb.org/3
* Auth: Append ?api\_key=YOUR\_API\_KEY

**4.] Service**

**auth.service.ts**

**Code to fetch Data from TMDB**

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

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class AuthService {

// private url = 'http://localhost:3000/api';

// private url = 'http://localhost:3000/api';

private apiKey = 'c41ed211b980b62273cd19473beca22c';

private apiUrl = 'https://api.themoviedb.org/3';

constructor(private http: HttpClient) { }

getMovieDetails(id: number): Observable<any> {

return this.http.get(

`${this.apiUrl}/movie/${id}?api\_key=${this.apiKey}`

);

}

getFeelGoodMovies(): Observable<any> {

return this.http.get(

`${this.apiUrl}/discover/movie?api\_key=${this.apiKey}&with\_genres=35,10751,18,10749&sort\_by=popularity.desc&language=en-US`

);

}

getActionFixMovies(): Observable<any> {

return this.http.get( `${this.apiUrl}/discover/movie?api\_key=${this.apiKey}&with\_genres=28&sort\_by=popularity.desc&language=en-US`

);

}

getMindBenderMovies(): Observable<any> {

return this.http.get( `${this.apiUrl}/discover/movie?api\_key=${this.apiKey}&with\_genres=53,9648,878,18&sort\_by=popularity.desc&language=en-US`

);

}

searchMovies(query: string): Observable<any> {

return this.http.get(`${this.apiUrl}/search/movie`, {

params: {

api\_key: this.apiKey,

query: query

}

});

}

}

**5.] Genre (TMDB Genre Ids)**

* **Feel Good**: use genres list 35,10751,18,10749
* **Action**: use genres list 28
* **Mind Bender**: use genres list 53,9648,878,18

**6.] Components**

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

import { AuthService } from '../auth.service';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { RouterModule ,Router } from '@angular/router';

@Component({

selector: 'app-feel-good',

standalone: true,

imports: [CommonModule, HttpClientModule , RouterModule],

templateUrl: './feel-good.component.html',

styleUrl: './feel-good.component.css'

})

export class FeelGoodComponent implements OnInit{

movies: any[] = [];

constructor(private authservice: AuthService) {}

ngOnInit() {

this.authservice.getFeelGoodMovies().subscribe((data) => {

this.movies = data.results;

}); }}

**7.] HTML Template**

<section class="feel\_good\_banner">

<div class="container">

<div class="card new\_card">

<div class="card-body">

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a [routerLink]="['/index']">Overview</a></li>

<li class="breadcrumb-item active" aria-current="page">feel good</li>

</ol>

</nav>

</div>

</div>

<h2>Feel Good Movies</h2>

<div class="row">

<div class="col-md-3" \*ngFor="let movie of movies">

<div class="card mb-3">

<img class="card-img-top" [src]="'https://image.tmdb.org/t/p/w500' + movie.poster\_path"

alt="{{ movie.title }}" />

<div class="card-body">

<h5 class="card-title">{{ movie.title }}</h5>

<!-- <p class="card-text">{{ movie.overview }}</p> -->

<a [routerLink]="['/movie', movie.id]" class="btn btn-click-me">Watch now</a>

</div>

</div>

</div>

</div>

</div>

</section>

**8.] Unit Testing**

import { TestBed } from '@angular/core/testing';

import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';

import { MovieService } from './movie.service';

import { environment } from '../../environments/environment';

describe('MovieService', () => {

let service: MovieService;

let httpMock: HttpTestingController;

beforeEach(() => {

TestBed.configureTestingModule({

imports: [HttpClientTestingModule],

providers: [MovieService],

});

service = TestBed.inject(MovieService);

httpMock = TestBed.inject(HttpTestingController);

});

it('should fetch movies by genre ID', () => {

const dummyResponse = { results: [{ title: 'Movie 1' }] };

service.getMoviesByGenre(28).subscribe((res) => {

expect(res.results.length).toBe(1);

expect(res.results[0].title).toBe('Movie 1');

});

const req = httpMock.expectOne(

`${service['baseUrl']}/discover/movie?api\_key=${environment.tmdbApiKey}&with\_genres=28`

);

expect(req.request.method).toBe('GET');

req.flush(dummyResponse);

});

afterEach(() => {

httpMock.verify();

});

});

**8.] Best Practice**

 Use **interfaces or models** for movie objects (type safety).

 Modularize: keep services and components loosely coupled.

 Cache genre results if needed to reduce API calls.

 Use Angular's async pipe where possible to reduce subscription boilerplate