A screenshot of a computer

AI-generated content may be incorrect.

import 'package:day\_7/screens/product\_details.dart';

import 'package:day\_7/web\_service/api\_service.dart';

import 'package:day\_7/model/movie.dart';

import 'package:day\_7/model/TvShow.dart';

import 'package:day\_7/model/product.dart';

import 'package:flutter/material.dart';

class TvShowslistpage extends StatefulWidget {

  const TvShowslistpage({super.key});

  @override

  State<TvShowslistpage> createState() => \_TvShowslistpageState();

}

// LifeCyclehooks

class \_TvShowslistpageState extends State<TvShowslistpage> {

  late Future<List<TvShow>> tvShow;

  final ScrollController \_scrollController = ScrollController();

  int currentPage = 1;

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    tvShow = ApiService().getTVshow(currentPage);

    \_scrollController.addListener(() {

      if (\_scrollController.position.pixels >=

          \_scrollController.position.maxScrollExtent) {

        getMoreTV();

      } else if (\_scrollController.position.pixels <=

          \_scrollController.position.minScrollExtent) {

        getBeforeTV();

      }

    });

  }

  void getMoreTV() {

    setState(() {

      currentPage++;

      tvShow = ApiService().getTVshow(currentPage);

    });

  }

  void getBeforeTV() {

    setState(() {

      if (currentPage > 1) {

        currentPage--;

        tvShow = ApiService().getTVshow(currentPage);

      }

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Colors.amber,

        title: Text('Tvshows Page'),

      ),

      body: FutureBuilder<List<TvShow>>(

        future: tvShow,

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return Center(

                child: CircularProgressIndicator(

              backgroundColor: Colors.blue,

              color: Colors.pink,

            ));

          } else if (snapshot.hasError) {

            return Center(

              child: Text('Exception ${snapshot.error}'),

            );

          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

            return Center(

              child: Text('No TV shows avaliable'),

            );

          }

          return ListView.builder(

            controller: \_scrollController,

            itemBuilder: (context, index) {

              final tv = snapshot.data![index];

              return GestureDetector(

                onTap: () {},

                child: Card(

                  margin: EdgeInsets.symmetric(vertical: 6, horizontal: 10),

                  shape: RoundedRectangleBorder(

                      borderRadius: BorderRadius.circular(10)),

                  child: Row(

                    children: [

                      Container(

                        width: 100,

                        height: 100,

                        decoration: BoxDecoration(

                            borderRadius: BorderRadius.circular(8),

                            image: DecorationImage(

                              image: NetworkImage(

                                  'https://image.tmdb.org/t/p/w500${tv.posterPath}'),

                              fit: BoxFit.cover,

                            )),

                      ),

                      SizedBox(

                        width: 5,

                      ),

                      Expanded(

                          child: Column(

                        crossAxisAlignment: CrossAxisAlignment.start,

                        children: [

                          Text(

                            tv.name,

                            maxLines: 1,

                            overflow: TextOverflow.ellipsis,

                            style: TextStyle(

                              fontSize: 16,

                              fontWeight: FontWeight.bold,

                            ),

                          ),

                          SizedBox(

                            height: 5,

                          ),

                          Text(

                            tv.overview,

                            maxLines: 1,

                            overflow: TextOverflow.ellipsis,

                            style: TextStyle(

                              fontSize: 14,

                              fontWeight: FontWeight.bold,

                            ),

                          )

                        ],

                      ))

                    ],

                  ),

                ),

              );

            },

            itemCount: snapshot.data!.length,

          );

        },

      ),

    );

  }

}

class TvShow {

final bool adult;

//final String backdropPath;

final List<int> genreIds;

final List<String> origincountry;

final int id;

final String originalLanguage;

final String originalTitle;

final String overview;

final double popularity;

final String posterPath;

final String firstairdate;

final String name;

final double voteAverage;

final int voteCount;

TvShow(

{required this.adult,

//required this.backdropPath,

required this.genreIds,

required this.origincountry,

required this.id,

required this.originalLanguage,

required this.originalTitle,

required this.overview,

required this.popularity,

required this.posterPath,

required this.firstairdate,

required this.name,

required this.voteAverage,

required this.voteCount});

factory TvShow.fromJson(Map<String, dynamic> json) {

return TvShow(

adult :json['adult'],

//backdropPath :json['backdrop\_path'],

genreIds : json['genre\_ids'].cast<int>(),

origincountry :json['origin\_country'].cast<String>(),

id : json['id'],

originalLanguage : json['original\_language'],

originalTitle : json['original\_name'],

overview : json['overview'],

popularity :json['popularity'],

posterPath :json['poster\_path'],

firstairdate: json['first\_air\_date'],

name : json['name'],

voteAverage : json['vote\_average'],

voteCount : json['vote\_count']);

}

}

Future<List<TvShow>> getTVshow(int pageNumber) async {

// Map<String,dynamic> data={

// 'movies':List<Movie>,

// 'pages':response.data['pages']

// }

try {

final response = await \_dio.get(

'https://api.themoviedb.org/3/tv/airing\_today?api\_key=e9d830a02f5c47e82cdedbcc8318ca4d&page=$pageNumber');

if (response.statusCode == 200) {

return (response.data['results'] as List)

.map((tvjson) => TvShow.fromJson(tvjson))

.toList();

} else {

throw Exception('Error casting data');

}

} catch (e) {

throw Exception('Error getting Data');

}

}