EXPERIMENT NO:2

Name:Mahi Jodhani
Roll no: 21
D15A
Aim: To design Flutter UI by including common widgets.
Theory:
1. Introduction

Flutter is an open-source UI development framework created by Google that enables developers to build cross-platform applications using a single codebase. It follows a widget-based architecture, where every UI element is represented as a widget. Widgets in Flutter are categorized into two types: StatelessWidgets (which do not change dynamically) and StatefulWidgets (which maintain dynamic states).

2. Objective

- Understand the role of widgets in Flutter UI development.
- Explore different types of common widgets and their functionalities.
- Learn how to structure a UI effectively using widgets.

3. Importance of Widgets in Flutter

- Composable – Multiple widgets can be combined to build a complete UI.
- Reusable – The same widget can be used in different parts of the application.
- Customizable – Widgets offer extensive customization options to achieve the desired look and feel.
- Responsive – Widgets adapt to different screen sizes, making the UI flexible across devices.

4. Commonly Used Widgets in Flutter

Flutter provides a wide range of widgets to design modern UIs.

1. Structural Widgets – Define the layout and structure of the UI (e.g., Container, Column, Row).
2. Interactive Widgets – Handle user input and interaction (e.g., Button, TextField, GestureDetector).
3. Styling Widgets – Enhance the visual appearance (e.g., Padding, Align, Card).
4. Scrolling Widgets – Enable scrolling functionality (e.g., ListView, GridView).

5. Implementation of UI in Flutter

Designing a UI in Flutter involves:

1. Defining the widget tree – A hierarchical arrangement of widgets that form the UI structure.
2. Using layout widgets – Arranging elements using Column, Row, Stack, and other layout-based widgets.
3. Adding interactivity – Incorporating buttons, text fields, and gesture detectors for user interaction.

4. Applying styling and theming – Customizing widgets with colors, padding, borders, and shadows to enhance aesthetics.

Code:
Main.Dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:netflix_clone/application/downloads/downloads_bloc.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/domain/core/di/injectable.dart';
import 'package:netflix_clone/presentation/main_page/widgets/screen_main_page.dart';
import 'package:netflix_clone/presentation/splash/screen_splash.dart';

import 'application/description/description_bloc.dart';
import 'application/fast_laugh/fast_laugh_bloc.dart';
import 'application/home/home_bloc.dart';
import 'application/hot_and_new/hot_and_new_bloc.dart';
import 'application/search/search_bloc.dart';
import 'presentation/onStartPage/scrren_onboarding.dart';

Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await configureInjection();
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
  return MultiBlocProvider(
   providers: [
    BlocProvider(
     create: (ctx) => getIt<DownloadsBloc>(),
    ),
    BlocProvider(
     create: (ctx) => getIt<SearchBloc>(),
    ),
    BlocProvider(
     create: (ctx) => getIt<FastLaughBloc>(),
    ),
    BlocProvider(
     create: (ctx) => getIt<HotAndNewBloc>(),
    ),
    BlocProvider(
     create: (ctx) => getIt<HomeBloc>(),
    ),
    BlocProvider(
     create: (ctx) => getIt<DescriptionBloc>(),
    )
   ],
   child: MaterialApp(
    title: 'Flutter Demo',
    theme: ThemeData(
        appBarTheme: const AppBarTheme(backgroundColor: Colors.black),
        primarySwatch: Colors.blue,
        backgroundColor: Colors.black,
        scaffoldBackgroundColor: backgrounndColor,
        fontFamily: GoogleFonts.montserrat().fontFamily,
        textTheme: const TextTheme(
            bodyText1: TextStyle(color: Colors.white),
            bodyText2: TextStyle(color: Colors.white))),
    home:  const ScreenSplash() ,
   ),
  );
 }
}
```

Screen_home.dart:

```
import 'dart:developer';
import 'dart:ffi';

import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:netflix_clone/application/home/home_bloc.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/home/widget/background_card.dart';
import 'package:netflix_clone/presentation/home/widget/custom_button_widget.dart';
import 'package:netflix_clone/presentation/home/widget/number_card.dart';
import 'package:netflix_clone/presentation/home/widget/number_title_card.dart';
import 'package:netflix_clone/presentation/widget/main_title.dart';
import 'package:netflix_clone/presentation/widget/main_title_card.dart';
```

```
// ignore: must_be_immutable
class ScreenHome extends StatelessWidget {
  ScreenHome({Key? key}) : super(key: key);

  ValueNotifier<bool> scrollNotifier = ValueNotifier(true);
  @override
  Widget build(BuildContext context) {
    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      BlocProvider.of<HomeBloc>(context).add(const GetHomeScreenData());
    });
    return Scaffold(
        body: ValueListenableBuilder(
            valueListenable: scrollNotifier,
            builder: (BuildContext context, index2, _) {
              return NotificationListener<UserScrollNotification>(
                onNotification: (notificaton) {
                  final ScrollDirection direction = notificaton.direction;
                  if (direction == ScrollDirection.reverse) {
                    scrollNotifier.value = false;
                  } else if (direction == ScrollDirection.forward) {
                    scrollNotifier.value = true;
                  }
                  return true;
                },
                child: RefreshIndicator(
                  onRefresh: () async {
                    BlocProvider.of<HomeBloc>(context)
                        .add(const GetHomeScreenData());
                  },
                  child: Stack(
                    children: [
                      BlocBuilder<HomeBloc, HomeState>(
                        builder: (context, state) {
                          if (state.isLoading) {
                            return const Center(
                              child: CircularProgressIndicator(
                                strokeWidth: 2,
                              ),
                            );
                          } else if (state.hasError) {
                            return const Center(
                                child: Text(
                              "error while getting data",
                              style: TextStyle(color: kwhiteColor),
  ));
}
// released past year
final _releasedPastYear =
    state.pastYearMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
// get id pase year
final _releasedPastYearId =
    state.pastYearMovieList.map((m) {
  return m.id;
}).toList();
// _releasedPastYearId.shuffle();
// trending
final _trending = state.trendingMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
_trending.shuffle();
// get id trendimg
final _trendingId = state.trendingMovieList.map((m) {
  return m.id;
}).toList();
_trendingId.shuffle();
//  trendse drama
final _trendse = state.tenseDramaMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
_trendse.shuffle();
// get id trendse
final _trendsId = state.tenseDramaMovieList.map((m) {
  return m.id;
}).toList();
// south indian movie
final _southIndia =
    state.southIndianMovieList.map((m) {
  return '$imageAppendUrl${m.posterPath}';
}).toList();
// get id pase year
final _southIndiaId =
    state.southIndianMovieList.map((m) {
  return m.id;
}).toList();
// tv shows
final _top10tvShows = state.trendingTvList.map((m) {
      return '$imageAppendUrl${m.posterPath}';
    }).toList();
    // get id pase year
    final _top10tvShowsId = state.trendingTvList.map((m) {
      return m.id;
    }).toList();
    print(state.trendingMovieList.length);
    return ListView(
      children: [
        const BackgroundCard(),
        kHeight,
        MainTitleCard(
          id: _releasedPastYearId,
          title: "Released in the past year",
          posterList: _releasedPastYear,
        ),
        kHeight,
        MainTitleCard(
          id: _trendingId,
          title: "Trending Now",
          posterList: _trending,
        ),
        kHeight,
        NumberTitleCard(
          posterList: _top10tvShows,
          title: "Top 10 TV shows in india today",
        ),
        kHeight,
        MainTitleCard(
          id: _trendsId,
          title: "Tense Dramas",
          posterList: _trendse,
        ),
        kHeight,
        MainTitleCard(
          id: _southIndiaId,
          title: "South Indian Cinema",
          posterList: _southIndia,
        ),
      ],
    );
  },
),
scrollNotifier.value == true
```

```
                    ? AnimatedContainer(
                        duration: const Duration(milliseconds: 1000),
                        width: double.infinity,
                        height: 90,
                        color: Colors.black.withOpacity(0.2),
                        child: Column(
                          children: [
                            Row(
                              children: [
                                Image.network(

'https://cdn-images-1.medium.com/max/1200/1*ty4NvNrGg4ReETxqU2N3Og.png',
                                  width: 60,
                                  height: 60,
                                ),
                                const Spacer(),
                                const Icon(
                                  Icons.cast,
                                  size: 30,
                                  color: kwhiteColor,
                                ),
                                kWidth,
                                Container(
                                  color: Colors.blue,
                                  width: 30,
                                  height: 30,
                                ),
                                kWidth
                              ],
                            ),
                            Row(
                              mainAxisAlignment:
                                  MainAxisAlignment.spaceEvenly,
                              children: const [
                                Text(
                                  "TV Shows",
                                  style: kHomeTitleText,
                                ),
                                Text(
                                  "Movies",
                                  style: kHomeTitleText,
                                ),
                                Text(
                                  "Categories",
                                        style: kHomeTitleText,
                                      )
                                    ],
                                  )
                                ],
                              ),
                            )
                          : kHeight
                    ],
                  ),
                ),
              );
            }));
  }
}
```

Main_cart.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:netflix_clone/core/constants.dart';
import '../../application/description/description_bloc.dart';
import '../decriptions/Screen_decription.dart';

class MainCard extends StatelessWidget {
  final String imageUrl;
  const MainCard({
    Key? key,
    required this.imageUrl,
    required this.id,
  }) : super(key: key);
  final int id;
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 10),
      child: GestureDetector(
        onTap: () {
          print("movie id $id");
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (ctx) => ScreenDescription(
                id: id,
              ),
            ),
          );
        },
        child: Container(
          width: 130,
          height: 250,
          decoration: BoxDecoration(
            borderRadius: kRadius10,
            image: DecorationImage(
              fit: BoxFit.fill,
              image: NetworkImage(imageUrl),
            ),
          ),
        ),
      ),
    );
  }
}
```

Main_title.dart:

```
import 'package:flutter/material.dart';

class MainTitle extends StatelessWidget {
 const MainTitle({Key? key, required this.title}) : super(key: key);

 final String title;
 @override
 Widget build(BuildContext context) {
  return Text(
   title,
   style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
  );
 }
}
```

Main_title_card.dart:

```
import 'package:flutter/material.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/widget/main_card.dart';
import 'package:netflix_clone/presentation/widget/main_title.dart';

class MainTitleCard extends StatelessWidget {
 const MainTitleCard({
   Key? key,
   required this.title,
   required this.posterList,
   required this.id,
 }) : super(key: key);
 final String title;
 final List<String?> posterList;
 final List<int?> id;
 @override
 Widget build(BuildContext context) {
   return Column(
     crossAxisAlignment: CrossAxisAlignment.start,
     children: [
       MainTitle(title: title),
       kHeight,
       LimitedBox(
         maxHeight: 200,
         child: ListView(
             scrollDirection: Axis.horizontal,
             children: List.generate(
                 posterList.length,
                 (index) => MainCard(
                       id: id[index]!,
                       imageUrl: posterList[index]!,
                     ))),
       )
     ],
   );
 }
}
```

Video_widget.dart:

```
import 'package:flutter/material.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';
import 'package:netflix_clone/presentation/search/widget/search_idel.dart';

class VideoWidget extends StatelessWidget {
  const VideoWidget({
    Key? key,
    required this.imageUrl,
  }) : super(key: key);

  final String imageUrl;
  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        SizedBox(
          width: double.infinity,
          height: 200,
          child: Image.network(
            imageUrl,
            fit: BoxFit.cover,
            loadingBuilder:
                (BuildContext _, Widget child, ImageChunkEvent? progress) {
              if (progress == null) {
                return child;
              } else {
                return const Center(
                    child: CircularProgressIndicator(
                  strokeWidth: 2,
                ));
              }
            },
            errorBuilder: (BuildContext _, Object a, StackTrace? trace) {
              return const Center(
                  child: Icon(
                Icons.wifi,
                color: kwhiteColor,
              ));
            },
          ),
        ),
      ],),}}
```
App_bar_widget.dart:
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:netflix_clone/core/colors/colors.dart';
import 'package:netflix_clone/core/constants.dart';

class AppBarWidget extends StatelessWidget {
  const AppBarWidget({Key? key, required this.title}) : super(key: key);
  final String title;
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        kWidth,
        Text(
          title,
          style: const TextStyle(fontSize: 30, fontWeight: FontWeight.bold),
        ),
        const Spacer(),
        const Icon(
          Icons.cast,
          size: 30,
          color: kwhiteColor,
        ),
        kWidth,
        Container(
          color: Colors.blue,
          width: 30,
          height: 30,
        ),
        kWidth
      ],
    );
  }
}
```

Home_bloc.dart:

```
import 'package:bloc/bloc.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
import 'package:injectable/injectable.dart';
import 'package:netflix_clone/domain/core/failures/main_failure.dart';
import 'package:netflix_clone/domain/new_and_hot/hot_and_new_service.dart';

import '../../domain/new_and_hot/model/discover.dart';

part 'home_event.dart';
part 'home_state.dart';
part 'home_bloc.freezed.dart';

@injectable
class HomeBloc extends Bloc<HomeEvent, HomeState> {
  final HotAndNewService _homeService;
```

```
HomeBloc(this._homeService) : super(HomeState.initial()) {
 // get home screendata
 on<GetHomeScreenData>((event, emit) async {
  // set loading to ui
  emit(state.copyWith(isLoading: true, hasError: false));
  //   // get datat
  final _movieResult = await _homeService.getHotAndNewMovieData();
  final _tvResutl = await _homeService.getHotAndNewTvData();

  //   //transform data
  final stateOne = _movieResult.fold((MainFailure failures) {
   return HomeState(
    stateId: DateTime.now().millisecondsSinceEpoch.toString(),
    pastYearMovieList: [],
    trendingMovieList: [],
    tenseDramaMovieList: [],
    southIndianMovieList: [],
    trendingTvList: [],
    isLoading: false,
    hasError: true,
   );
  }, (HotAndNewDataResp resp) {
   final pastYear = resp.results;
   pastYear.shuffle();
   final trending = resp.results;
   trending.shuffle();
   final tenseDarama = resp.results;
   tenseDarama.shuffle();
   final southIndia = resp.results;
   southIndia.shuffle();
   return HomeState(
    stateId: DateTime.now().millisecondsSinceEpoch.toString(),
    pastYearMovieList: pastYear,
    trendingMovieList: trending,
    tenseDramaMovieList: tenseDarama,
    southIndianMovieList: southIndia,
    trendingTvList: state.trendingMovieList,
    isLoading: false,
    hasError: false,
   );
  });
  emit(stateOne);
  final stateTwo = _tvResutl.fold((MainFailure failure) {
   return HomeState(
          stateId: DateTime.now().millisecondsSinceEpoch.toString(),
          pastYearMovieList: [],
          trendingMovieList: [],
          tenseDramaMovieList: [],
          southIndianMovieList: [],
          trendingTvList: [],
          isLoading: false,
          hasError: true,
        );
      }, (HotAndNewDataResp resp) {
        final topTenList = resp.results;
        return HomeState(
          stateId: DateTime.now().millisecondsSinceEpoch.toString(),
          pastYearMovieList: state.pastYearMovieList,
          trendingMovieList: state.trendingMovieList,
          tenseDramaMovieList: state.tenseDramaMovieList,
          southIndianMovieList: state.southIndianMovieList,
          trendingTvList: topTenList,
          isLoading: false,
          hasError: false,
        );
      });
      emit(stateTwo);
    });
  }
}
```

Home_bloc.freezed.dart:

```
part of 'home_bloc.dart';

// **************************************************************************
// FreezedGenerator
// **************************************************************************

T _$identity<T>(T value) => value;

final _privateConstructorUsedError = UnsupportedError(
    'It seems like you constructed your class using `MyClass._()`. This constructor is only meant to be used by freezed and you are not supposed to need it nor use it.\nPlease check the documentation here for more information: https://github.com/rrousselGit/freezed#custom-getters-and-methods');

/// @nodoc
mixin _$HomeEvent {
  @optionalTypeArgs
  TResult when<TResult extends Object?>({
    required TResult Function() getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult? whenOrNull<TResult extends Object?>({
    TResult Function()? getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult maybeWhen<TResult extends Object?>({
    TResult Function()? getHomeScreenData,
    required TResult orElse(),
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult map<TResult extends Object?>({
    required TResult Function(GetHomeScreenData value) getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult? mapOrNull<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
  }) =>
      throw _privateConstructorUsedError;
  @optionalTypeArgs
  TResult maybeMap<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
    required TResult orElse(),
  }) =>
      throw _privateConstructorUsedError;
}

/// @nodoc
abstract class $HomeEventCopyWith<$Res> {
  factory $HomeEventCopyWith(HomeEvent value, $Res Function(HomeEvent) then) =
      _$HomeEventCopyWithImpl<$Res>;
}

/// @nodoc
class _$HomeEventCopyWithImpl<$Res> implements $HomeEventCopyWith<$Res> {
  _$HomeEventCopyWithImpl(this._value, this._then);

  final HomeEvent _value;
  // ignore: unused_field
  final $Res Function(HomeEvent) _then;
}

/// @nodoc
abstract class _$$GetHomeScreenDataCopyWith<$Res> {
  factory _$$GetHomeScreenDataCopyWith(
          _$GetHomeScreenData value, $Res Function(_$GetHomeScreenData) then) =
      __$$GetHomeScreenDataCopyWithImpl<$Res>;
}

/// @nodoc
class __$$GetHomeScreenDataCopyWithImpl<$Res>
    extends _$HomeEventCopyWithImpl<$Res>
    implements _$$GetHomeScreenDataCopyWith<$Res> {
  __$$GetHomeScreenDataCopyWithImpl(
      _$GetHomeScreenData _value, $Res Function(_$GetHomeScreenData) _then)
      : super(_value, (v) => _then(v as _$GetHomeScreenData));

  @override
  _$GetHomeScreenData get _value => super._value as _$GetHomeScreenData;
}

/// @nodoc

class _$GetHomeScreenData implements GetHomeScreenData {
  const _$GetHomeScreenData();

  @override
  String toString() {
    return 'HomeEvent.getHomeScreenData()';
  }

  @override
  bool operator ==(dynamic other) {
    return identical(this, other) ||
        (other.runtimeType == runtimeType && other is _$GetHomeScreenData);
  }

  @override
  int get hashCode => runtimeType.hashCode;

  @override
  @optionalTypeArgs
```

```
TResult when<TResult extends Object?>({
  required TResult Function() getHomeScreenData,
}) {
  return getHomeScreenData();
}

@override
@optionalTypeArgs
TResult? whenOrNull<TResult extends Object?>({
  TResult Function()? getHomeScreenData,
}) {
  return getHomeScreenData?.call();
}

@override
@optionalTypeArgs
TResult maybeWhen<TResult extends Object?>({
  TResult Function()? getHomeScreenData,
  required TResult orElse(),
}) {
  if (getHomeScreenData != null) {
    return getHomeScreenData();
  }
  return orElse();
}

@override
@optionalTypeArgs
TResult map<TResult extends Object?>({
  required TResult Function(GetHomeScreenData value) getHomeScreenData,
}) {
  return getHomeScreenData(this);
}

@override
@optionalTypeArgs
TResult? mapOrNull<TResult extends Object?>({
  TResult Function(GetHomeScreenData value)? getHomeScreenData,
}) {
  return getHomeScreenData?.call(this);
}

@override
@optionalTypeArgs
```

```
  TResult maybeMap<TResult extends Object?>({
    TResult Function(GetHomeScreenData value)? getHomeScreenData,
    required TResult orElse(),
  }) {
    if (getHomeScreenData != null) {
      return getHomeScreenData(this);
    }
    return orElse();
  }
}

abstract class GetHomeScreenData implements HomeEvent {
  const factory GetHomeScreenData() = _$GetHomeScreenData;
}

/// @nodoc
mixin _$HomeState {
  String get stateId => throw _privateConstructorUsedError;
  List<HotAndNewData> get pastYearMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get trendingMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get tenseDramaMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get southIndianMovieList =>
      throw _privateConstructorUsedError;
  List<HotAndNewData> get trendingTvList => throw _privateConstructorUsedError;
  bool get isLoading => throw _privateConstructorUsedError;
  bool get hasError => throw _privateConstructorUsedError;

  @JsonKey(ignore: true)
  $HomeStateCopyWith<HomeState> get copyWith =>
      throw _privateConstructorUsedError;
}

/// @nodoc
abstract class $HomeStateCopyWith<$Res> {
  factory $HomeStateCopyWith(HomeState value, $Res Function(HomeState) then) =
      _$HomeStateCopyWithImpl<$Res>;
  $Res call(
      {String stateId,
      List<HotAndNewData> pastYearMovieList,
      List<HotAndNewData> trendingMovieList,
      List<HotAndNewData> tenseDramaMovieList,
      List<HotAndNewData> southIndianMovieList,
      List<HotAndNewData> trendingTvList,
      bool isLoading,
      bool hasError});
}

/// @nodoc
class _$HomeStateCopyWithImpl<$Res> implements $HomeStateCopyWith<$Res> {
  _$HomeStateCopyWithImpl(this._value, this._then);

  final HomeState _value;
  // ignore: unused_field
  final $Res Function(HomeState) _then;

  @override
  $Res call({
    Object? stateId = freezed,
    Object? pastYearMovieList = freezed,
    Object? trendingMovieList = freezed,
    Object? tenseDramaMovieList = freezed,
    Object? southIndianMovieList = freezed,
    Object? trendingTvList = freezed,
    Object? isLoading = freezed,
    Object? hasError = freezed,
  }) {
    return _then(_value.copyWith(
      stateId: stateId == freezed
          ? _value.stateId
          : stateId // ignore: cast_nullable_to_non_nullable
              as String,
      pastYearMovieList: pastYearMovieList == freezed
          ? _value.pastYearMovieList
          : pastYearMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      trendingMovieList: trendingMovieList == freezed
          ? _value.trendingMovieList
          : trendingMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      tenseDramaMovieList: tenseDramaMovieList == freezed
          ? _value.tenseDramaMovieList
          : tenseDramaMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      southIndianMovieList: southIndianMovieList == freezed
          ? _value.southIndianMovieList
          : southIndianMovieList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      trendingTvList: trendingTvList == freezed
          ? _value.trendingTvList
          : trendingTvList // ignore: cast_nullable_to_non_nullable
              as List<HotAndNewData>,
      isLoading: isLoading == freezed
          ? _value.isLoading
          : isLoading // ignore: cast_nullable_to_non_nullable
              as bool,
      hasError: hasError == freezed
          ? _value.hasError
          : hasError // ignore: cast_nullable_to_non_nullable
              as bool,
    ));
  }
}

/// @nodoc
abstract class _$$_InitialCopyWith<$Res> implements $HomeStateCopyWith<$Res> {
  factory _$$_InitialCopyWith(
          _$_Initial value, $Res Function(_$_Initial) then) =
      __$$_InitialCopyWithImpl<$Res>;
  @override
  $Res call(
      {String stateId,
      List<HotAndNewData> pastYearMovieList,
      List<HotAndNewData> trendingMovieList,
      List<HotAndNewData> tenseDramaMovieList,
      List<HotAndNewData> southIndianMovieList,
      List<HotAndNewData> trendingTvList,
      bool isLoading,
      bool hasError});
}

/// @nodoc
class __$$_InitialCopyWithImpl<$Res> extends _$HomeStateCopyWithImpl<$Res>
    implements _$$_InitialCopyWith<$Res> {
  __$$_InitialCopyWithImpl(_$_Initial _value, $Res Function(_$_Initial) _then)
      : super(_value, (v) => _then(v as _$_Initial));

  @override
  _$_Initial get _value => super._value as _$_Initial;
```

```
@override
$Res call({
  Object? stateId = freezed,
  Object? pastYearMovieList = freezed,
  Object? trendingMovieList = freezed,
  Object? tenseDramaMovieList = freezed,
  Object? southIndianMovieList = freezed,
  Object? trendingTvList = freezed,
  Object? isLoading = freezed,
  Object? hasError = freezed,
}) {
  return _then(_$_Initial(
    stateId: stateId == freezed
        ? _value.stateId
        : stateId // ignore: cast_nullable_to_non_nullable
            as String,
    pastYearMovieList: pastYearMovieList == freezed
        ? _value._pastYearMovieList
        : pastYearMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    trendingMovieList: trendingMovieList == freezed
        ? _value._trendingMovieList
        : trendingMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    tenseDramaMovieList: tenseDramaMovieList == freezed
        ? _value._tenseDramaMovieList
        : tenseDramaMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    southIndianMovieList: southIndianMovieList == freezed
        ? _value._southIndianMovieList
        : southIndianMovieList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    trendingTvList: trendingTvList == freezed
        ? _value._trendingTvList
        : trendingTvList // ignore: cast_nullable_to_non_nullable
            as List<HotAndNewData>,
    isLoading: isLoading == freezed
        ? _value.isLoading
        : isLoading // ignore: cast_nullable_to_non_nullable
            as bool,
    hasError: hasError == freezed
        ? _value.hasError
        : hasError // ignore: cast_nullable_to_non_nullable
            as bool,
    ));
  }
}

/// @nodoc

class _$_Initial implements _Initial {
  const _$_Initial(
      {required this.stateId,
      required final List<HotAndNewData> pastYearMovieList,
      required final List<HotAndNewData> trendingMovieList,
      required final List<HotAndNewData> tenseDramaMovieList,
      required final List<HotAndNewData> southIndianMovieList,
      required final List<HotAndNewData> trendingTvList,
      required this.isLoading,
      required this.hasError})
      : _pastYearMovieList = pastYearMovieList,
        _trendingMovieList = trendingMovieList,
        _tenseDramaMovieList = tenseDramaMovieList,
        _southIndianMovieList = southIndianMovieList,
        _trendingTvList = trendingTvList;

  @override
  final String stateId;
  final List<HotAndNewData> _pastYearMovieList;
  @override
  List<HotAndNewData> get pastYearMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_pastYearMovieList);
  }

  final List<HotAndNewData> _trendingMovieList;
  @override
  List<HotAndNewData> get trendingMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_trendingMovieList);
  }

  final List<HotAndNewData> _tenseDramaMovieList;
  @override
  List<HotAndNewData> get tenseDramaMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_tenseDramaMovieList);
  }
  final List<HotAndNewData> _southIndianMovieList;
  @override
  List<HotAndNewData> get southIndianMovieList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_southIndianMovieList);
  }

  final List<HotAndNewData> _trendingTvList;
  @override
  List<HotAndNewData> get trendingTvList {
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_trendingTvList);
  }

  @override
  final bool isLoading;
  @override
  final bool hasError;

  @override
  String toString() {
    return 'HomeState(stateId: $stateId, pastYearMovieList: $pastYearMovieList, trendingMovieList:
$trendingMovieList, tenseDramaMovieList: $tenseDramaMovieList, southIndianMovieList:
$southIndianMovieList, trendingTvList: $trendingTvList, isLoading: $isLoading, hasError: $hasError)';
  }

  @override
  bool operator ==(dynamic other) {
    return identical(this, other) ||
        (other.runtimeType == runtimeType &&
            other is _$_Initial &&
            const DeepCollectionEquality().equals(other.stateId, stateId) &&
            const DeepCollectionEquality()
                .equals(other._pastYearMovieList, _pastYearMovieList) &&
            const DeepCollectionEquality()
                .equals(other._trendingMovieList, _trendingMovieList) &&
            const DeepCollectionEquality()
                .equals(other._tenseDramaMovieList, _tenseDramaMovieList) &&
            const DeepCollectionEquality()
                .equals(other._southIndianMovieList, _southIndianMovieList) &&
            const DeepCollectionEquality()
                .equals(other._trendingTvList, _trendingTvList) &&
            const DeepCollectionEquality().equals(other.isLoading, isLoading) &&
            const DeepCollectionEquality().equals(other.hasError, hasError));
  }

  @override
  int get hashCode => Object.hash(
      runtimeType,
      const DeepCollectionEquality().hash(stateId),
      const DeepCollectionEquality().hash(_pastYearMovieList),
      const DeepCollectionEquality().hash(_trendingMovieList),
      const DeepCollectionEquality().hash(_tenseDramaMovieList),
      const DeepCollectionEquality().hash(_southIndianMovieList),
      const DeepCollectionEquality().hash(_trendingTvList),
      const DeepCollectionEquality().hash(isLoading),
      const DeepCollectionEquality().hash(hasError));

  @JsonKey(ignore: true)
  @override
  _$$_InitialCopyWith<_$_Initial> get copyWith =>
      __$$_InitialCopyWithImpl<_$_Initial>(this, _$identity);
}

abstract class _Initial implements HomeState {
  const factory _Initial(
      {required final String stateId,
      required final List<HotAndNewData> pastYearMovieList,
      required final List<HotAndNewData> trendingMovieList,
      required final List<HotAndNewData> tenseDramaMovieList,
      required final List<HotAndNewData> southIndianMovieList,
      required final List<HotAndNewData> trendingTvList,
      required final bool isLoading,
      required final bool hasError}) = _$_Initial;

  @override
  String get stateId;
  @override
  List<HotAndNewData> get pastYearMovieList;
  @override
  List<HotAndNewData> get trendingMovieList;
  @override
  List<HotAndNewData> get tenseDramaMovieList;
  @override
  List<HotAndNewData> get southIndianMovieList;
  @override
  List<HotAndNewData> get trendingTvList;
  @override
  bool get isLoading;
  @override
  bool get hasError;
  @override
  @JsonKey(ignore: true)
  _$$_InitialCopyWith<_$_Initial> get copyWith =>
      throw _privateConstructorUsedError;
}
```

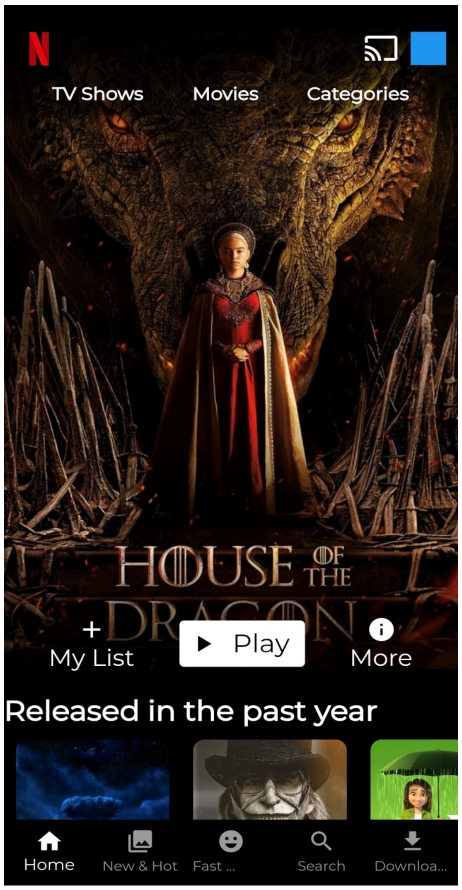
Home_event.dart:

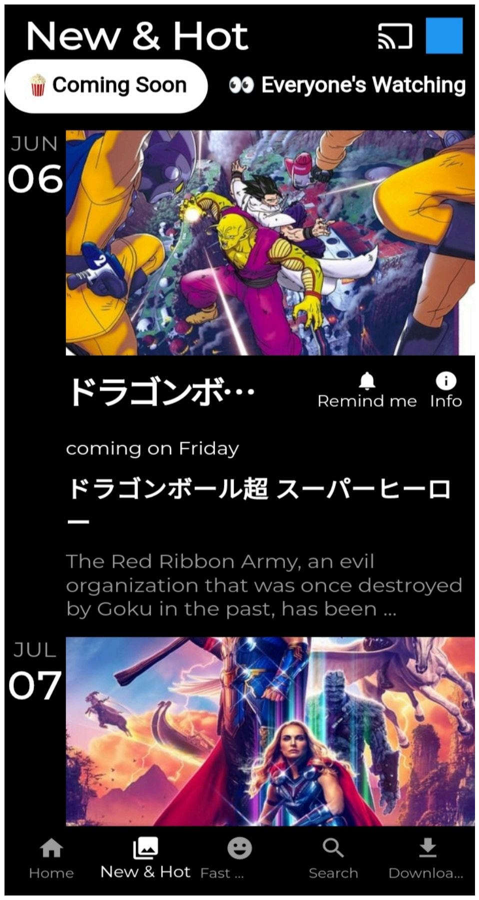
```
part of 'home_bloc.dart';

@freezed
class HomeEvent with _$HomeEvent {
  const factory HomeEvent.getHomeScreenData() = GetHomeScreenData;
}
```

Home_state.dart:

```
part of 'home_bloc.dart';

@freezed
class HomeState with _$HomeState {
  const factory HomeState(
      {required String stateId,
      required List<HotAndNewData> pastYearMovieList,
      required List<HotAndNewData> trendingMovieList,
      required List<HotAndNewData> tenseDramaMovieList,
      required List<HotAndNewData> southIndianMovieList,
      required List<HotAndNewData> trendingTvList,
      required bool isLoading,
      required bool hasError}) = _Initial;
  factory HomeState.initial() => const HomeState(
        stateId: '0',
        pastYearMovieList: [],
        trendingMovieList: [],
        tenseDramaMovieList: [],
        southIndianMovieList: [],
        trendingTvList: [],
        isLoading: false,
        hasError: false,
      );
}
```

Screenshots:





coming on Friday

ドラゴンボール超 スーパーヒーロー

The Red Ribbon Army, an evil organization that was once destroyed by Goku in the past, has been ...

Conclusion

Flutter’s widget-based design pattern makes UI development intuitive and flexible. By utilizing common widgets effectively, developers can create seamless, responsive, and visually engaging user interfaces. This experiment provides a foundation for understanding Flutter’s UI structure and the role of widgets in application development.