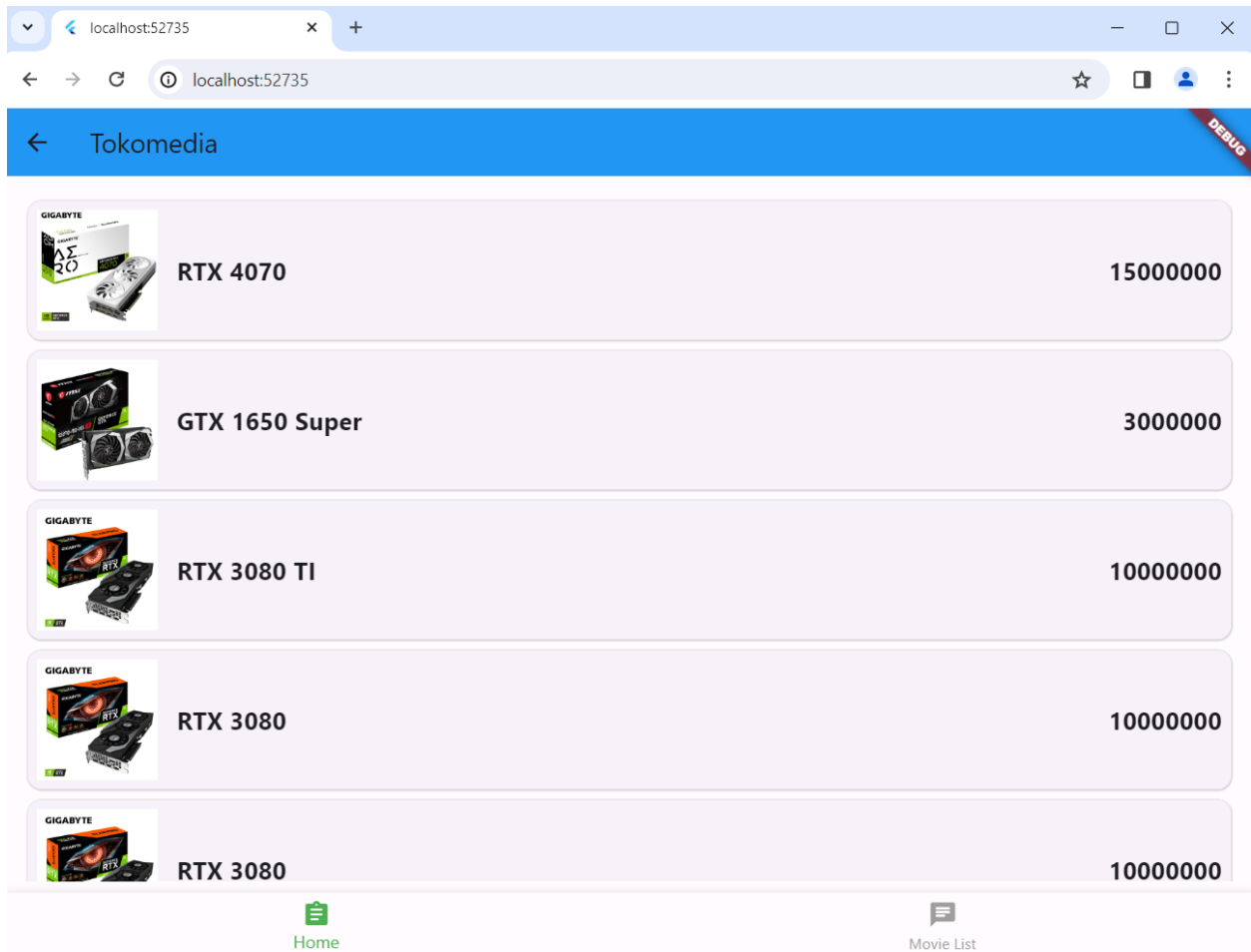


Nama : Putra Hermawan Ramadhani

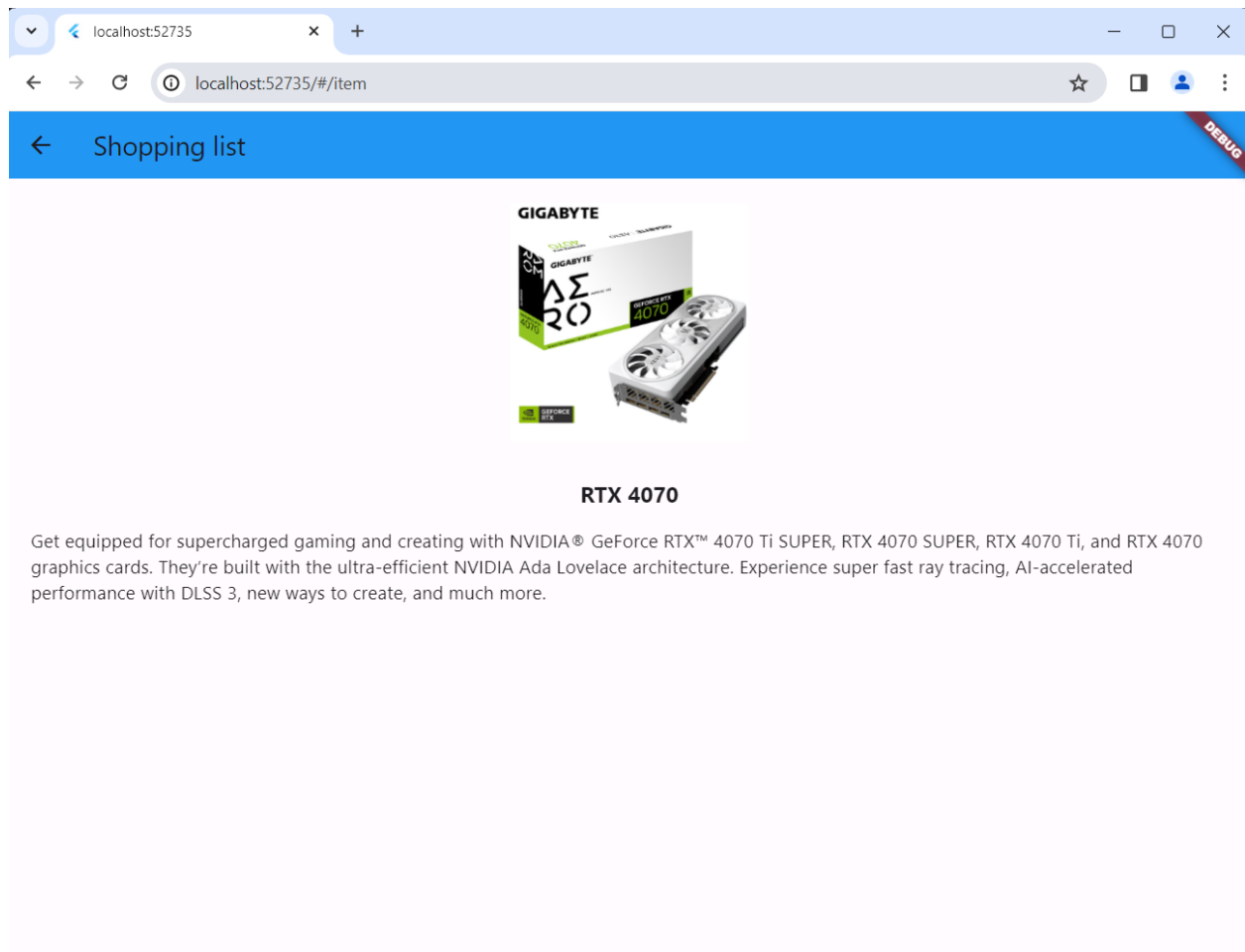
Kelas/Absen : XI RPL4 / 32

Screenshoot

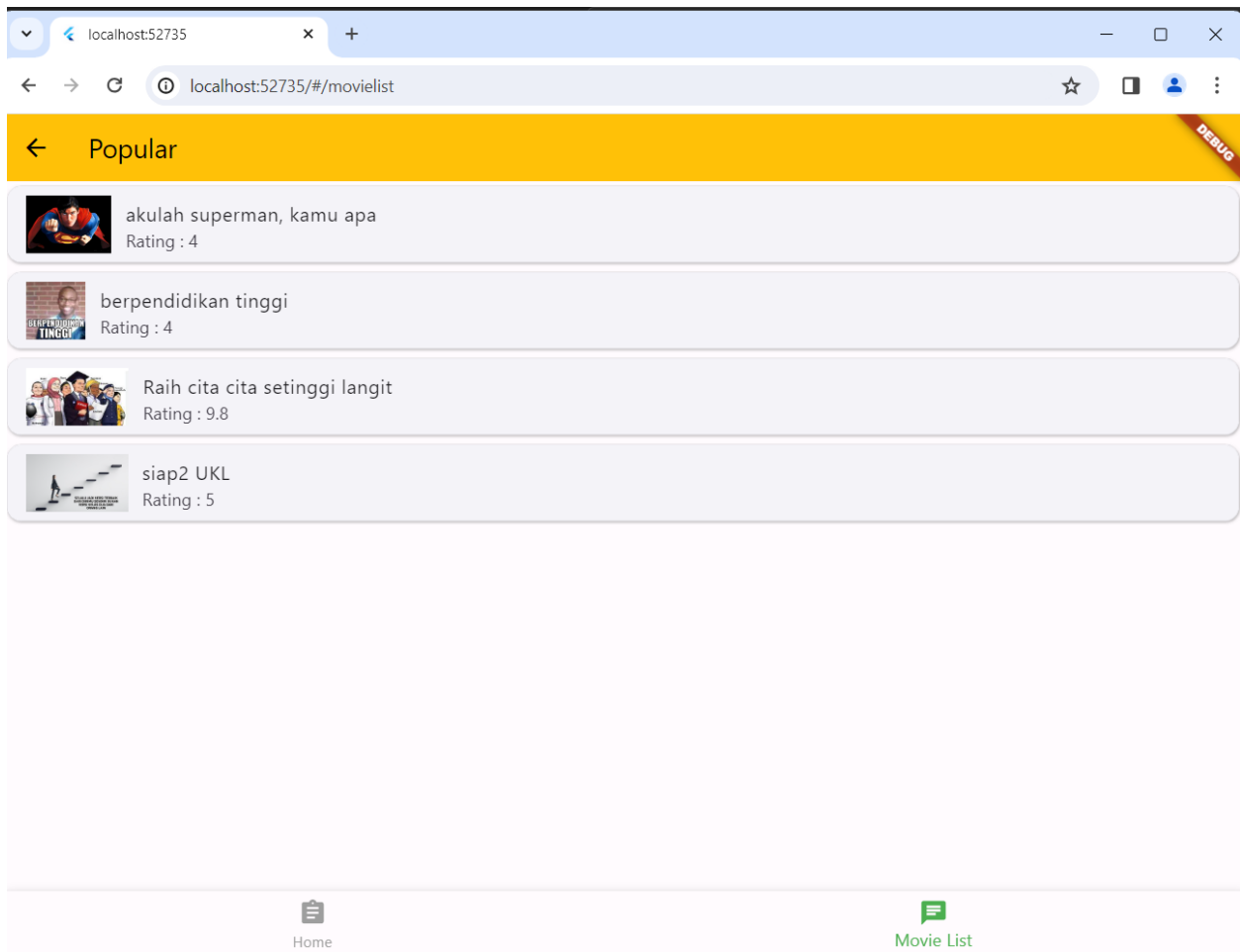
Layar 1 :



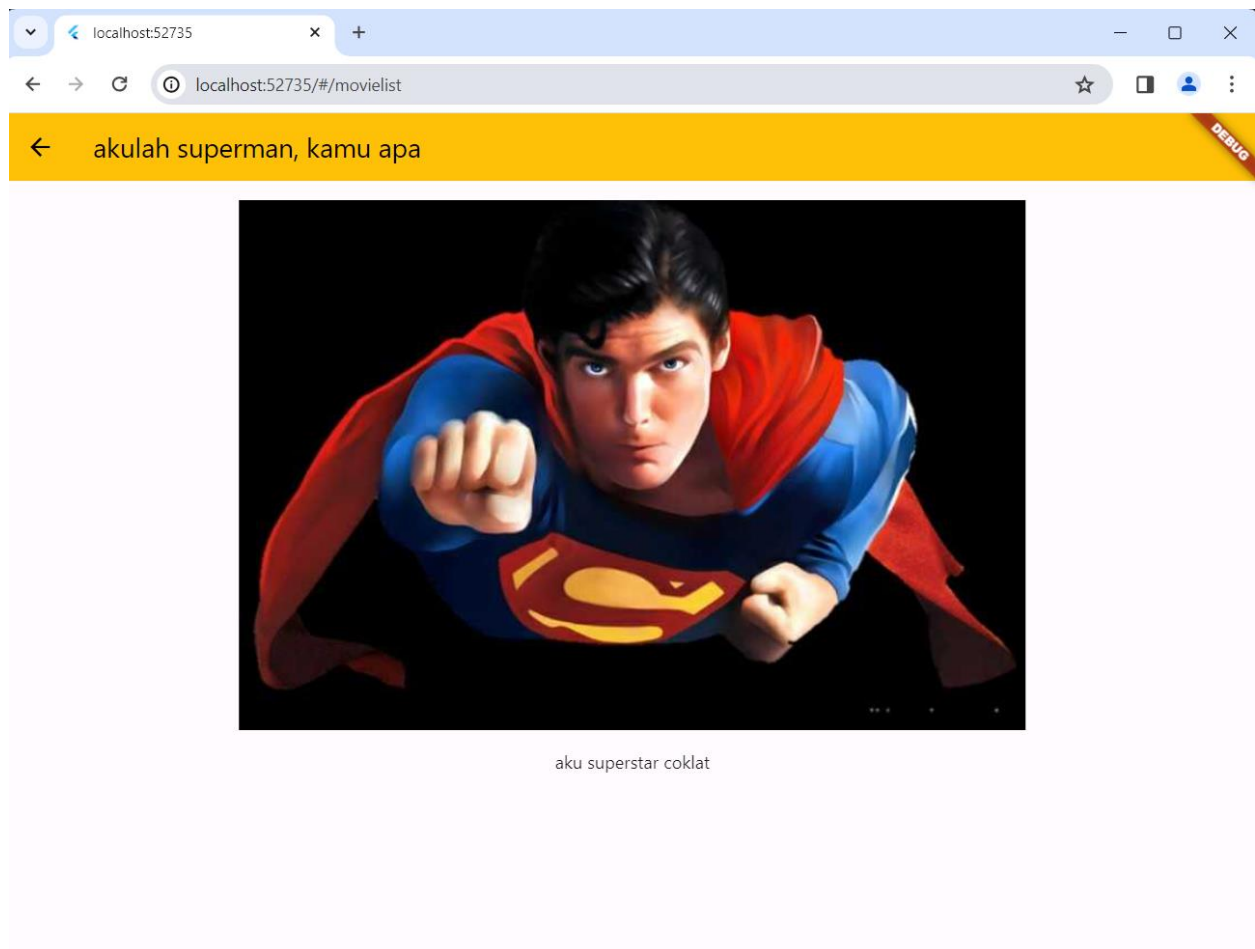
Layar 1 Bagian 2 :



Layar 2 bagian 1 :



Layar 2 bagian 2 :



Penjelasan Script :

```

class HttpService {
    final String baseUrl = 'https://movie.tukanginyuk.com/api/getmovie';

    Future<List?> getPopularMovies() async {
        final String uri = baseUrl;

        http.Response result = await http.get(Uri.parse(uri));

        if(result.statusCode == HttpStatus.ok){
            print("Connected");
            final jsonResponse = json.decode(result.body);
            final moviesMap = jsonResponse['data'];
            List movie = moviesMap.map((i) => Film.fromJson(i)).toList();
            return movie;
        }else{
            print("Not Connected");
            return null;
        }
    }
}

```

Penjelasan :

Deklarasi class `HttpService`: Ini adalah deklarasi awal dari class yang akan digunakan untuk mengelola permintaan HTTP.

1. Variabel `baseUrl`: Ini adalah URL dasar dari API yang akan digunakan untuk mengambil data film. URL ini diinisialisasi dengan nilai `'https://movie.tukanginyuk.com/api/getmovie'`.
2. Method `getPopularMovies()`: Ini adalah method untuk mengambil daftar film populer dari API. Method ini mengembalikan sebuah objek `Future` yang akan berisi daftar film atau `null` jika terjadi kesalahan.

- Method ini didefinisikan sebagai `async` sehingga dapat menggunakan `await` untuk menunggu hasil dari operasi yang asinkron.
- Variabel `uri`: Variabel ini digunakan untuk menyimpan URL lengkap untuk permintaan HTTP, yang sama dengan `baseUrl` dalam kasus ini.
- Permintaan HTTP dilakukan menggunakan `http.get()`, dimana URL yang diminta di-parse dengan `Uri.parse(uri)`. Ini akan mengembalikan objek `Response` yang berisi respon dari permintaan HTTP.

- Setelah mendapatkan respon, dilakukan pemeriksaan pada status kode respon. Jika status kode adalah `HttpStatus.ok` (200), maka data dianggap berhasil diambil.
- Di dalam blok `if`, respon dari server di-decode dari JSON ke objek Dart menggunakan `json.decode(result.body)`. Ini akan menghasilkan `Map<String, dynamic>` dari data JSON.
 - Data film yang diterima dari API kemudian diambil dari kunci `'data'` dalam objek JSON dan diubah menjadi daftar film menggunakan metode `map()` dan `toList()`. Setiap item dalam daftar film akan diubah menjadi objek Dart `Film` menggunakan konstruktor `Film.fromJson(i)` yang diasumsikan telah ada.
 - Akhirnya, daftar film yang dihasilkan akan dikembalikan.
- Jika status kode tidak `HttpStatus.ok`, maka akan dicetak pesan "Not Connected" dan nilai `null` akan dikembalikan, menandakan bahwa tidak ada data yang berhasil diambil.

```
class Film {
  int? id;
  String? title;
  double? voteAverage;
  String? overview;
  String? posterPath;

  Film(this.id, this.title, this.voteAverage, this.overview, this.posterPath);

  Film.fromJson(Map<String, dynamic> parsedJson){
    id = parsedJson['id'];
    title = parsedJson['title'];
    voteAverage = parsedJson['voteaverage']*1.0;
    overview = parsedJson['overview'];
    posterPath = parsedJson['posterpath'];
  }
}
```

Penjelasan :

1. Variabel-instance `id`, `title`, `voteAverage`, `overview`, dan `posterPath`: Ini adalah properti-properti dari class `Film`. Mereka memiliki tipe data nullable (ditandai dengan `?` setelah tipe data), yang berarti mereka bisa memiliki nilai `null`.
2. Konstruktor `Film`: Ini adalah konstruktor yang digunakan untuk membuat objek `Film`. Konstruktor ini mengambil lima parameter: `id`, `title`, `voteAverage`, `overview`, dan `posterPath`. Parameter-parameter ini kemudian digunakan untuk menginisialisasi properti-properti dari objek `Film`.
3. Metode `Film.fromJson()`: Ini adalah metode yang digunakan untuk membuat objek `Film` dari data JSON. Metode ini mengambil satu parameter, yaitu `parsedJson`, yang merupakan `Map<String, dynamic>` yang berisi data JSON yang akan diubah menjadi objek `Film`.

- Di dalam metode ini, nilai-nilai dari `parsedJson` diambil berdasarkan kunci yang sesuai (seperti `'id'`, `'title'`, dll.) dan disimpan ke dalam properti-properti yang sesuai dari objek `Film`.
- Terdapat penanganan khusus untuk `voteAverage`, dimana nilai tersebut dikonversi menjadi tipe data `double` dan kemudian dikalikan dengan 1.0. Hal ini mungkin dilakukan untuk memastikan bahwa `voteAverage` memiliki tipe data `double` dan bukan `int`.

```
class item {  
  String name;  
  int price;  
  String imagePath;  
  String description;  
  item({required this.name, required this.price, required this.imagePath, required th  
}
```

1. Penjelasan :
Variabel-instance:

- `name`: Variabel bertipe `String` yang menyimpan nama barang.
- `price`: Variabel bertipe `int` yang menyimpan harga barang.

- **ImagePath**: Variabel bertipe **String** yang menyimpan path atau lokasi dari gambar barang.
- **description**: Variabel bertipe **String** yang menyimpan deskripsi singkat tentang barang tersebut.

2. Konstruktor:

- Konstruktor **Item** memiliki parameter bertipe named parameter yang dibuat dengan menggunakan curly braces (**{}**).
- Parameter-parameter konstruktor ini adalah **name**, **price**, **ImagePath**, dan **description**.
- Semua parameter tersebut dinyatakan sebagai wajib (required) menggunakan kata kunci **required**, yang berarti setiap kali objek **Item** dibuat, nilai untuk parameter-parameter tersebut harus disediakan.
- Ketika objek **Item** dibuat, nilai-nilai dari parameter yang diberikan akan disalin ke dalam properti-properti yang sesuai dalam objek **Item**.


```

class itempage extends StatelessWidget {
  itempage({super.key});

  @override
  Widget build(BuildContext context) {
    final itemArgs = (ModalRoute.of(context)!.settings.arguments) as Map;
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.blue,
        title: const Text('Shopping list'),
      ), // AppBar
      body: Container(
        padding: EdgeInsets.all(20),
        child: Column(
          children: [
            Container(
              width: 200, // Set the desired width for the image
              height: 200,
              child: Image.network(itemArgs["ImagePath"]),
            ), // Container
            SizedBox(
              height: 30), // Add spacing between image and description // SizedBox
            Container(
              child: Text(
                itemArgs["name"],
                style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold // Set the font size for the description
                ), // TextStyle
              ), // Text
            ), // Container
            SizedBox(height: 16), // Add spacing between image and name
            Text(
              itemArgs["description"],
              style: TextStyle(
                fontSize: 15, // Set the font size to make the name bigger
              ), // TextStyle
            ), // Text
          ],
        ), // Column
      ), // Container
    ); // Scaffold
  }
}

```

Penjelasan :

Secara keseluruhan, kode ini membuat daftar kartu, di mana setiap kartu menampilkan gambar dan beberapa teks. Ketika kartu diklik, aplikasi akan melakukan navigasi ke halaman lain dan mengirimkan data item sebagai argumen. Di bagian bawah, ada navigasi bar yang memungkinkan pengguna untuk beralih antara berbagai bagian dari aplikasi.

```

@override
void initState(){
  service = HttpService();
  initialize();
  super.initState();
}

@override
Widget build(BuildContext context) {
  service!.getPopularMovies().then((value) => {
    setState(() {
      result =(value != null) as String;
    })
  });
  return Scaffold(
    appBar: AppBar(
      title: const Text("Popular"),
      backgroundColor: Colors.amber,
      foregroundColor: Colors.black,
    ), // AppBar
    body: film!.isEmpty ? Container() :
    ListView.builder(
      itemCount: moviesCount,
      itemBuilder: (context, int position){
        return Card(
          color: Colors.white,
          elevation: 2.0,
          child: ListTile(
            leading: Image(image: NetworkImage(film![position].posterPath)),
            title: Text(film![position].title),
            subtitle: Text("Rating : ${film![position].voteAverage.toString()}"),
            onTap: (){
              MaterialPageRoute route = MaterialPageRoute(
                builder: (_) => MovieDetail(film?[position])); // MaterialPageRoute
              Navigator.push(context, route);
            },
          ), // ListTile
        ); // Card
      }, // ListView.builder
      bottomNavigationBar: BottomNav(selectedItem: 1)
    ); // Scaffold
  }
}

```

Penjelasan :

Kode ini menunjukkan penanganan data asinkron, pembuatan UI dengan widget, dan interaksi dengan layanan eksternal untuk pengambilan data.

```

class BottomNav extends StatefulWidget {
  final int selectedItem;
  BottomNav({super.key, required this.selectedItem});

  @override
  State<BottomNav> createState() => _BottomNavState();
}

class _BottomNavState extends State<BottomNav> {
  int selectedNavbar = 0;
  void changeSelectedNavBar(int index) {
    setState(() {
      selectedNavbar = index;
    });
    if (index == 0) {
      Navigator.pushNamed(context, '/');
    } else if (index == 1) {
      Navigator.pushNamed(context, '/movielist');
    }
  }
}

@override
Widget build(BuildContext context) {
  return BottomNavigationBar(
    items: const <BottomNavigationBarItem>[
      BottomNavigationBarItem(
        icon: Icon(Icons.assignment),
        label: 'Home',
      ), // BottomNavigationBarItem
      BottomNavigationBarItem(
        icon: Icon(Icons.chat),
        label: 'Movie List',
      ), // BottomNavigationBarItem
    ], // <BottomNavigationBarItem>[]
    selectedItemColor: Colors.green,
    unselectedItemColor: Colors.grey,
    showUnselectedLabels: true,
    currentIndex: widget.selectedItem,
    onTap: changeSelectedNavBar,
  ); // BottomNavigationBar
}
}

```

Penjelasan :

kode ini diatas memungkinkan pengguna untuk berpindah antara berbagai layar dalam aplikasi dengan mengklik item navbarlayar.

```

class MovieDetail extends StatelessWidget {
  final Film film;
  const MovieDetail(this.film);

  @override
  Widget build(BuildContext context) {
    String path;
    if (film.posterPath != null){
      path = film.posterPath!;
    } else {
      path = 'https://cdn.vectorstock.com/i/preview-1x/82/99/no-image-available-like-missing-picture-vector-43938299.jpg';
    }
    double height = MediaQuery.of(context).size.height;
    return Scaffold(
      appBar: AppBar(
        title: Text(film.title!),
        backgroundColor: Colors.amber,
        foregroundColor: Colors.black,
      ), // AppBar
      body: SingleChildScrollView(
        child: Center(
          child: Column(
            children: [
              Container(
                padding: EdgeInsets.all(16),
                height: height / 1.5,
                child: Image.network(path),
              ), // Container
              Container(
                child: Text(film.overview!),
                padding: EdgeInsets.only(left: 16, right: 16),
              ) // Container
            ],
          ), // Column
        ), // Center
      ), // SingleChildScrollView
    ); // Scaffold
  }
}

```

Penjelasan:

Kode tersebut bertujuan untuk mengganti gambar dengan gambar yang berada di path jika gagal load

```

Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => Homepage(),
      '/item': (context) => itempage(),
      '/movielist': (context) => FilmList(),
    },
  )); // MaterialApp
}

```

Penjelasan :

Kode ini menunjukkan bagaimana route dari aplikasi flutterApi ini.