1. Buatlah sebuah halaman baru yang memiliki spesifikasi berikut:
2. Memiliki appbar dengan judul “Contacts”
3. memilki icon (gambar bebas)
4. memiliki deskripsi (text bebas)
5. Buatlah sebuah Title yang bertuliskan “Create New Contacts”
6. menampilkan sebuah form input nama dan no telp
7. menampilkan tombol submit

* ListContactsModel.dart

import 'dart:ui';

class ListContactsModel {

String nama;

String no\_telp;

DateTime date;

Color? color;

String? image;

ListContactsModel({

required this.nama,

required this.no\_telp,

required this.date,

this.image,

this.color,

});

}

final List<ListContactsModel> contacts = [

ListContactsModel(

nama: "Name 1", no\_telp: "082228905435", date: DateTime(2023, 3, 24)),

ListContactsModel(

nama: "Name 2", no\_telp: "082228902464", date: DateTime(2023, 3, 24)),

];

* Contacts.dart

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:flutter\_platform\_widget/FlutterForm/ListContactsModel.dart';

import 'package:flutter\_colorpicker/flutter\_colorpicker.dart';

import 'package:file\_picker/file\_picker.dart';

import 'package:intl/intl.dart';

class Contacts extends StatefulWidget {

const Contacts({Key? key}) : super(key: key);

@override

State<Contacts> createState() => \_Contacts();

}

class \_Contacts extends State<Contacts> {

final \_formKey = GlobalKey<FormState>();

ListContactsModel? contactsModel;

final TextEditingController namaController = TextEditingController();

final TextEditingController nomorController = TextEditingController();

DateTime \_selectedDate = DateTime.now();

Color pickerColor = const Color(0xffEADDFF);

Color currentColor = Colors.blue;

String? \_fileName;

String \_filePath = "";

Future<void> \_selectDate(BuildContext context) async {

final DateTime? picked = await showDatePicker(

context: context,

initialDate: \_selectedDate,

firstDate: DateTime(1900),

lastDate: DateTime(2100),

);

if (picked != null && picked != \_selectedDate) {

setState(() {

\_selectedDate = picked;

});

}

}

void changeColor(Color color) {

setState(() => pickerColor = color);

}

void onColorChanged(Color color) {

setState(() => currentColor = color);

}

void \_openFilePicker() async {

try {

FilePickerResult? result = await FilePicker.platform.pickFiles(

type: FileType.any,

allowMultiple: false,

);

if (result != null) {

setState(() {

\_fileName = result.files.single.name;

\_filePath = result.files.first.path!;

});

}

} catch (e) {

print("Error while picking the file: " + e.toString());

}

}

@override

Widget build(BuildContext context) {

return Theme(

data: ThemeData(brightness: Brightness.light),

child: Scaffold(

appBar: AppBar(

backgroundColor: const Color(0xff6750A4),

title: const Text("Contacts"),

),

body: ListView(

children: <Widget>[

Container(

alignment: Alignment.center,

padding: const EdgeInsets.only(top: 81),

child: const Icon(Icons.contact\_page),

),

Container(

alignment: Alignment.center,

padding: const EdgeInsets.only(top: 10),

child: const Text(

"Create New Contacts",

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 24,

color: Color(0xff1C1B1F)),

),

),

Container(

alignment: Alignment.center,

padding: const EdgeInsets.only(top: 10),

margin: const EdgeInsets.only(left: 40, right: 40),

child: const Text(

"A dialog is a type of modal window that appears in front of app content to provide critical information, or prompt for a decision to be made.",

maxLines: 3,

overflow: TextOverflow.ellipsis,

textAlign: TextAlign.justify,

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 14,

color: Color(0xff49454F),

),

),

),

SizedBox(

height: 35.0,

child: Center(

child: Container(

padding: const EdgeInsets.only(top: 30),

margin:

const EdgeInsetsDirectional.only(start: 40.0, end: 40.0),

height: 1.0,

color: Colors.black12,

),

),

),

Form(

key: \_formKey,

child: Column(

children: [

Container(

padding: const EdgeInsets.only(top: 8),

margin: const EdgeInsets.only(left: 16.0, right: 16.0),

decoration: const BoxDecoration(

color: Color(0xffE7E0EC),

),

child: TextFormField(

controller: namaController,

validator: (value) {

List list = value!.split(' ');

if (value.isEmpty) {

return 'Nama harus di isi';

} else if (list.length < 2) {

return 'Nama harus terdiri dari minimal 2 kata';

} else if (!RegExp(

r'^[A-Z][a-zA-Z]\*(?: [A-Z][a-zA-Z]\*)\*$')

.hasMatch(value)) {

return 'Nama harus dimulai dengan huruf kapital';

} else if (!RegExp(r'^[a-zA-Z ]+$').hasMatch(value)) {

return 'Nama tidak boleh mengandung angka atau karakter khusus';

}

return null;

},

decoration: const InputDecoration(

floatingLabelBehavior: FloatingLabelBehavior.always,

contentPadding: EdgeInsets.only(left: 16),

hintText: "Insert Your Name",

hintStyle: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

),

labelText: "Name",

labelStyle: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 12,

),

),

),

),

Container(

padding: const EdgeInsets.only(top: 8),

margin: const EdgeInsets.only(

top: 15.0, left: 16.0, right: 16.0),

decoration: const BoxDecoration(

color: Color(0xffE7E0EC),

),

child: TextFormField(

// keyboardType: TextInputType.phone,

controller: nomorController,

validator: (value) {

if (value!.isEmpty) {

return 'Nomor harus di isi';

} else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {

return 'Nomor telepon harus terdiri dari angka saja';

} else if (value.length < 8) {

return 'Nomor telepon minimal 8 digit';

} else if (value.length > 15) {

return 'Nomor telepon maksimal 15 digit';

} else if (value[0] != '0') {

return 'Nomor telepon harus dimulai dengan angka 0';

}

return null;

},

decoration: const InputDecoration(

floatingLabelBehavior: FloatingLabelBehavior.always,

contentPadding: EdgeInsets.only(left: 16),

hintText: "+62 ....",

hintStyle: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

),

labelText: "Nomor",

labelStyle: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 12,

),

),

),

),

],

),

),

ListTile(

contentPadding:

const EdgeInsets.only(top: 10, left: 16, right: 16),

leading: const Text(

'Date',

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff1C1B1F),

fontWeight: FontWeight.bold,

),

),

trailing: TextButton(

onPressed: () => \_selectDate(context),

child: const Text(

'Select',

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

fontWeight: FontWeight.bold,

),

),

),

),

Container(

width: double.infinity,

padding: const EdgeInsets.only(left: 16),

child: Text(

'${\_selectedDate.day}-${\_selectedDate.month}-${\_selectedDate.year}',

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 14,

color: Color(0xff49454F),

),

),

),

const SizedBox(height: 20.0),

Container(

width: double.infinity,

padding: const EdgeInsets.only(left: 16),

child: const Text(

'Color',

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff1C1B1F),

fontWeight: FontWeight.bold),

),

),

const SizedBox(height: 10.0),

Container(

width: double.infinity,

height: 50.0,

margin: const EdgeInsets.only(left: 16, right: 16),

decoration: BoxDecoration(

color: currentColor,

shape: BoxShape.rectangle,

borderRadius: BorderRadius.circular(5),

),

),

const SizedBox(height: 20.0),

Container(

margin: const EdgeInsets.only(left: 130, right: 130),

child: RaisedButton(

color: currentColor,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(5),

),

onPressed: () {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: const Text('Pick a color'),

content: SingleChildScrollView(

child: ColorPicker(

pickerColor: pickerColor,

onColorChanged: changeColor,

showLabel: true,

pickerAreaHeightPercent: 0.8,

),

),

actions: [

FlatButton(

child: const Text('Cancel'),

onPressed: () {

Navigator.of(context).pop();

},

),

FlatButton(

child: const Text('Apply'),

onPressed: () {

onColorChanged(pickerColor);

Navigator.of(context).pop();

},

),

],

);

},

);

},

child: const Text(

'Pick Color',

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Colors.white,

),

),

),

),

const SizedBox(height: 20),

Container(

width: double.infinity,

padding: const EdgeInsets.only(left: 16),

child: const Text(

"Pick Files",

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

fontWeight: FontWeight.bold,

color: Color(0xff1C1B1F),

),

),

),

Container(

padding: const EdgeInsets.only(left: 16, top: 10),

child: Text(

\_fileName ?? "No file selected.",

),

),

\_filePath != "" ? Image.file(File(\_filePath)) : Container(),

Container(

margin: const EdgeInsets.only(left: 130, right: 130, top: 10),

child: RaisedButton(

onPressed: \_openFilePicker,

color: Colors.blue,

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(5),

),

child: const Text(

"Pick and Open File",

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Colors.white,

),

),

),

),

Container(

alignment: Alignment.centerRight,

padding: const EdgeInsets.only(top: 15, right: 15),

child: SizedBox(

width: 94,

height: 40,

child: ElevatedButton(

style: TextButton.styleFrom(

backgroundColor: const Color(0xff6750A4),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(20.0))),

onPressed: () {

if (\_formKey.currentState!.validate()) {

if (contactsModel.runtimeType != Null) {

ListContactsModel items = contacts.firstWhere(

(e) => e.no\_telp == contactsModel!.no\_telp);

items.nama = namaController.text;

items.no\_telp = nomorController.text;

items.date = \_selectedDate;

items.color = pickerColor;

items.image = \_filePath;

namaController.text = '';

nomorController.text = '';

\_fileName = null;

\_filePath = "";

currentColor = Colors.blue;

contactsModel = null;

setState(() {});

for (var item in contacts) {

print("${item.nama} | ${item.no\_telp}");

}

} else {

contacts.add(

ListContactsModel(

nama: namaController.text,

no\_telp: nomorController.text,

date: \_selectedDate,

color: pickerColor,

image: \_filePath,

),

);

for (var item in contacts) {

print("${item.nama} | ${item.no\_telp}");

}

namaController.text = '';

nomorController.text = '';

\_fileName = null;

\_filePath = "";

currentColor = Colors.blue;

setState(() {});

}

}

},

child: const Text(

"Submit",

style: TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 14,

),

),

),

),

),

Container(

alignment: Alignment.center,

padding: const EdgeInsets.only(top: 59),

child: const Text(

"List Contacts",

style: TextStyle(

// fontWeight: FontWeight.bold,

fontFamily: 'Roboto-Regular',

fontSize: 24,

color: Color(0xff1C1B1F),

),

),

),

Padding(

padding: const EdgeInsets.only(top: 10, bottom: 10),

child: ListView.separated(

shrinkWrap: true,

physics: const NeverScrollableScrollPhysics(),

itemBuilder: (ctx, i) {

return ListTile(

leading: Container(

clipBehavior: Clip.hardEdge,

decoration: BoxDecoration(

color: contacts[i].color.runtimeType != Null

? contacts[i].color

: const Color(0xffEADDFF),

shape: BoxShape.circle,

),

width: 50,

height: 50,

child: contacts[i].image.runtimeType != Null &&

contacts[i].image != ""

? Image.file(

File(contacts[i].image!),

fit: BoxFit.cover,

)

: Center(

child: Text(

contacts[i].nama[0],

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff21005D),

),

),

),

),

title: Text(

contacts[i].nama,

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff1C1B1F),

),

),

subtitle: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

contacts[i].no\_telp,

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 14,

color: Color(0xff49454F),

),

),

Text(DateFormat('dd-MM-yyyy')

.format(contacts[i].date)),

],

),

trailing: Wrap(

spacing: 23,

children: [

InkWell(

onTap: () {

setState(() {

contactsModel = contacts[i];

namaController.text = contacts[i].nama;

nomorController.text = contacts[i].no\_telp;

if (contacts[i].color.runtimeType != Null) {

currentColor = contacts[i].color!;

} else {

currentColor = Colors.blue;

}

if (contacts[i].image.runtimeType != Null &&

contacts[i].image != "") {

\_fileName = contacts[i].image!.split('/')[6];

\_filePath = contacts[i].image!;

} else {

\_fileName = null;

\_filePath = "";

}

});

},

child: const Icon(

Icons.edit\_outlined,

color: Color(0xff1C1B1F),

),

),

InkWell(

onTap: () {

contacts.removeAt(i);

setState(() {});

},

child: const Icon(

Icons.delete\_outline,

color: Color(0xff1C1B1F),

),

),

],

),

);

},

separatorBuilder: (ctx, i) {

return const Divider();

},

itemCount: contacts.length),

),

],

),

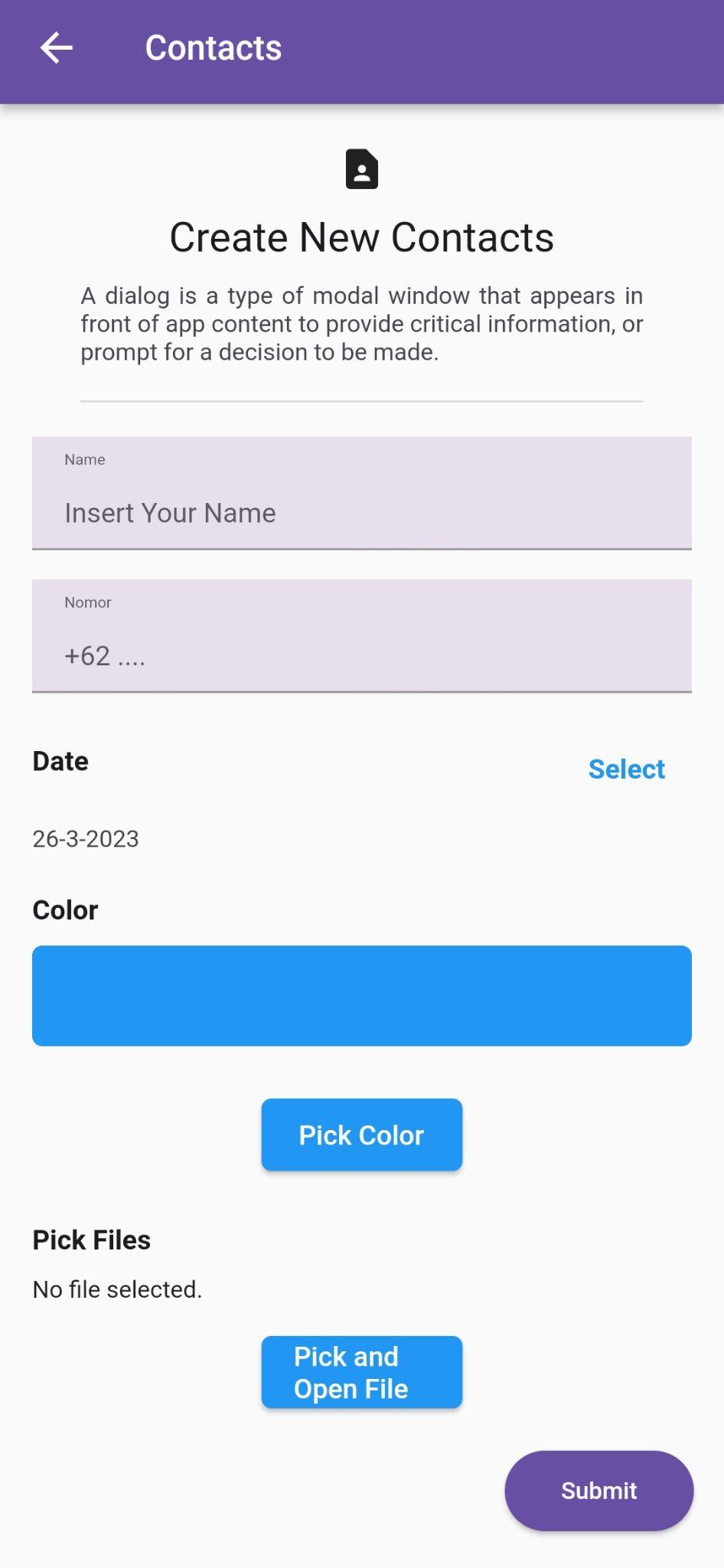
),

);

}

}

Output :



Penjelasan :

Untuk mengakses halaman ini dapat melalui drawers screen lalu klik Flutter Form. Dan pada halaman ini terdapat button kembali yang berfungsi untuk kembali ke tampilan drawers screen.Pada halaman ini berisikan form untuk menginputkan nama, nomor, date, color, file dan terdapat button submit.

1. Buatlah sebuah widget baru yang terletak dibawah form “create new contacts” dengan spesifikasi berikut:
2. Memiliki title “List Contacts”
3. menampilkkan data nama dan no telp dalam listview
4. Jika tombol submit di tekan maka data pada “List Contracts” akan bertambah

Code :

Container(

alignment: Alignment.center,

padding: const EdgeInsets.only(top: 59),

child: const Text(

"List Contacts",

style: TextStyle(

// fontWeight: FontWeight.bold,

fontFamily: 'Roboto-Regular',

fontSize: 24,

color: Color(0xff1C1B1F),

),

),

),

Padding(

padding: const EdgeInsets.only(top: 10, bottom: 10),

child: ListView.separated(

shrinkWrap: true,

physics: const NeverScrollableScrollPhysics(),

itemBuilder: (ctx, i) {

return ListTile(

leading: Container(

clipBehavior: Clip.hardEdge,

decoration: BoxDecoration(

color: contacts[i].color.runtimeType != Null

? contacts[i].color

: const Color(0xffEADDFF),

shape: BoxShape.circle,

),

width: 50,

height: 50,

child: contacts[i].image.runtimeType != Null &&

contacts[i].image != ""

? Image.file(

File(contacts[i].image!),

fit: BoxFit.cover,

)

: Center(

child: Text(

contacts[i].nama[0],

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff21005D),

),

),

),

),

title: Text(

contacts[i].nama,

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 16,

color: Color(0xff1C1B1F),

),

),

subtitle: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

contacts[i].no\_telp,

style: const TextStyle(

fontFamily: 'Roboto-Regular',

fontSize: 14,

color: Color(0xff49454F),

),

),

Text(DateFormat('dd-MM-yyyy')

.format(contacts[i].date)),

],

),

trailing: Wrap(

spacing: 23,

children: [

InkWell(

onTap: () {

setState(() {

contactsModel = contacts[i];

namaController.text = contacts[i].nama;

nomorController.text = contacts[i].no\_telp;

if (contacts[i].color.runtimeType != Null) {

currentColor = contacts[i].color!;

} else {

currentColor = Colors.blue;

}

if (contacts[i].image.runtimeType != Null &&

contacts[i].image != "") {

\_fileName = contacts[i].image!.split('/')[6];

\_filePath = contacts[i].image!;

} else {

\_fileName = null;

\_filePath = "";

}

});

},

child: const Icon(

Icons.edit\_outlined,

color: Color(0xff1C1B1F),

),

),

InkWell(

onTap: () {

contacts.removeAt(i);

setState(() {});

},

child: const Icon(

Icons.delete\_outline,

color: Color(0xff1C1B1F),

),

),

],

),

);

},

separatorBuilder: (ctx, i) {

return const Divider();

},

itemCount: contacts.length),

),

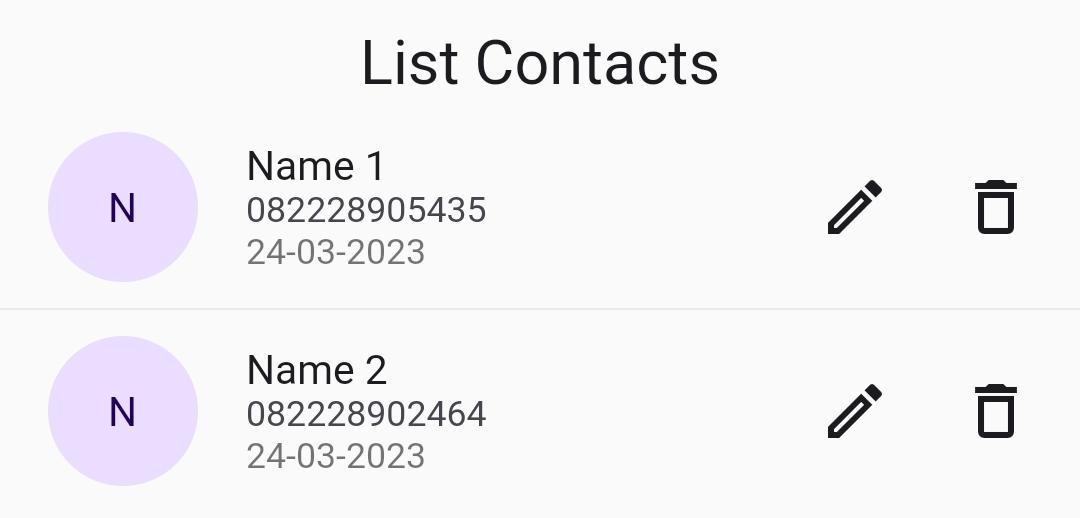
],

),

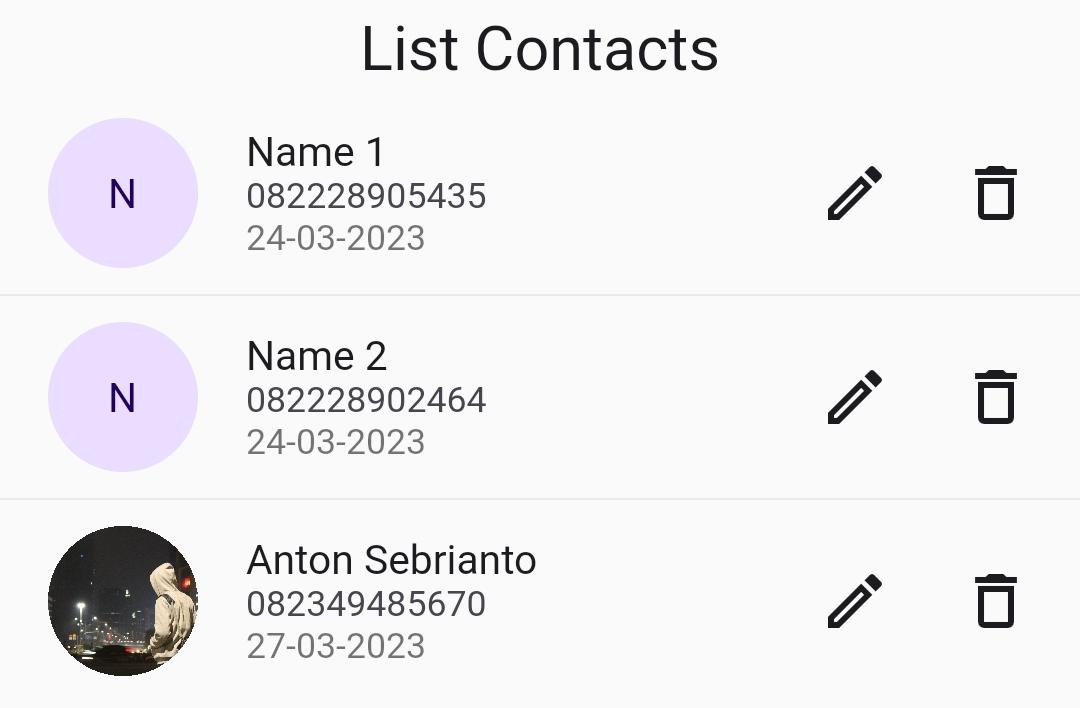
),

Output :

* Data List Contacts Sebelum di tambah (Data ini hanya sesuai dengan model yang telah dibuat).



* Data List Contacts setelah di tambah.



Penjelasan :

Data list contacts ini sesuai dengan data data yang telah dibuat pada ListContactsModel.dart yang telah saya tampilkan diatas pada nomor 1. Pada list contacts ini terdapat profile image, nama contacts, nomor contacts, date contacts, tombol edit untuk mengupdate data contacts, dan tombol hapus untuk menghapus contacts.