5 DECEMBER 2020

PET'S POINT- PANACEA TO ALL PET NEEDS

PROJECT REPORT

DESIGNED AND DEVELOPED BY

-YOGESH PARAB

S.K.SOMAIYA DEGREE COLLEGE OF ARTS, SCIENCE AND COMMERCE

VIDYAVIHAR(EAST)
MUMBAI-400 077

ACKNOWLEDGEMENT

I have a great pleasure in representing this project report entitled “Pet’s point- panacea to all pet needs” and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I would like to thank our honourable **Principal Dr. MANALI LONDHE** for granting us different facilities to do the project under the guidance of our faculty. Because to their support this project was a success.

I take this opportunity to thank **Prof. Mrs. Poonam Pandey**, Coordinator of the Department and all the professors of the Department of Computer Science of S.K.Somaiya Degree College of Arts Science & Commerce, for giving me an opportunity to complete this project and the most needed guidance throughout the duration of the Programme.

I am extremely grateful to my project guide **Prof. Poonam Pandey** for her valuable guidance and necessary support during each phase of the project. She was the source of continuous encouragement as each milestone was crossed.

A special thanks to the University Of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Sincere thanks from,

(YOGESH PARAB)

INDEX

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons.

-Yogesh Parab

APPLICATION OVERVIEW

Pet's Point is a one stop solution for all individuals pet related needs. The application helps you to keep up with all your pet demands and its requirements.An interactive mobile pet application available for use on both iOS and android platforms.The app lets you create, edit and set reminders for your pet. Also option to adopt and put up pets for adoption is available.
Pet's point tries to achieve the goal of bringing together people who are willing to help and nurture animals and the rescuing community by providing them a platform to interact

KEY FEATURES:

- Set feeding and grooming reminders
- Easy navigation and clean ui
- Google maps support for navigation to parks, vets etc
- Tips and tricks to train your pet more efficiently
- Calculate pet's BMI and keep track of his health
- Adopt a pet
- Contact/Help rescuers and NGO's
- Know more about various animal breeds

HARDWARE AND SOFTWARE REQUIREMENT:

- Android lollipop 5.0 and above
- Minimum 2 gb ram
- IOS 8 or newer

DEVELOPMENT TOOLS:

- IDE- Visual Studio
- Flutter SDK
- Ui Design - Adobe XD (View Design)

ABOUT FLUTTER:

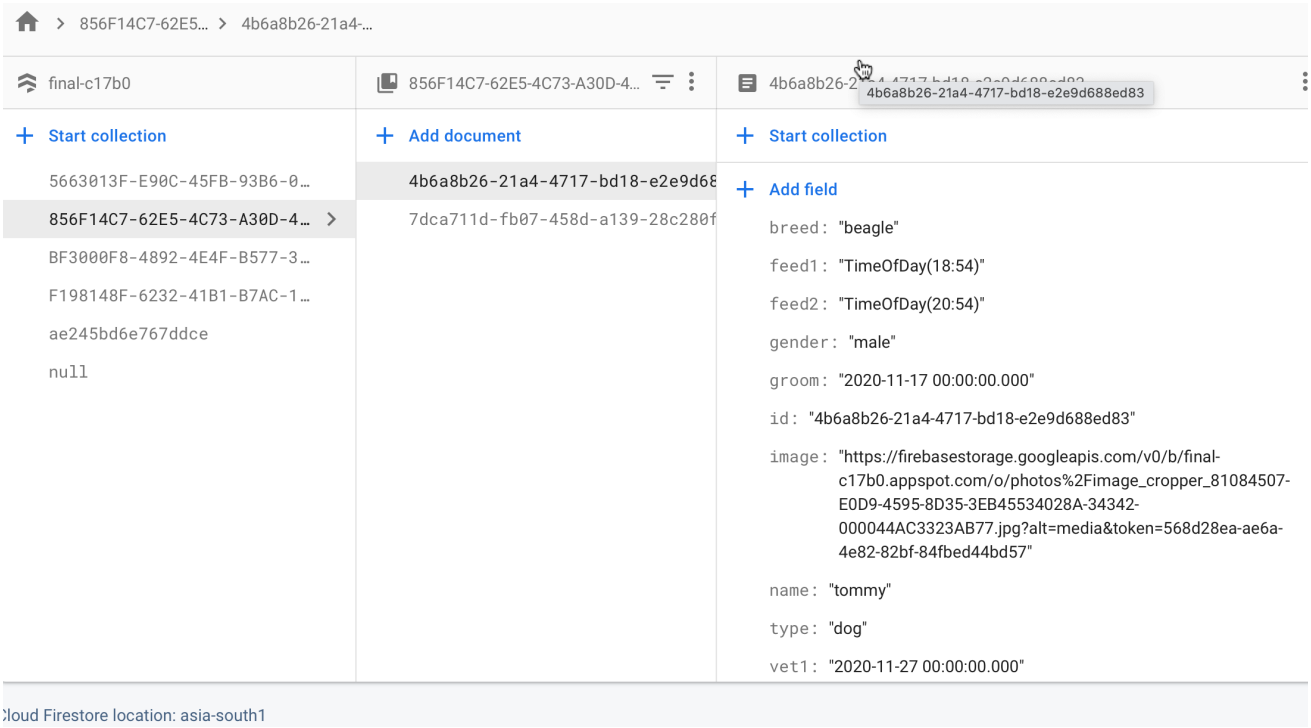
Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.

DATABASE DESIGN

- Used Firebase as backend database to store pets and their related details.Google **Firebase** is a Google-backed application development software that enables developers to develop iOS, Android and Web apps.

- Data-fields are stored in Firestore whereas images are stored inside firebase storage.Links to those images are fetched and stored in Firestore for access.

- User authentication is not implemented using Firestore authentication as the app does not have any practical implementation of storing user details. In order to provide personalised experience, the user tables are made unique by naming them by the device's IMEI code, thus making it unique to every user. Only shortcoming faced was, logging in on a new device would lead to a fresh copy of app.

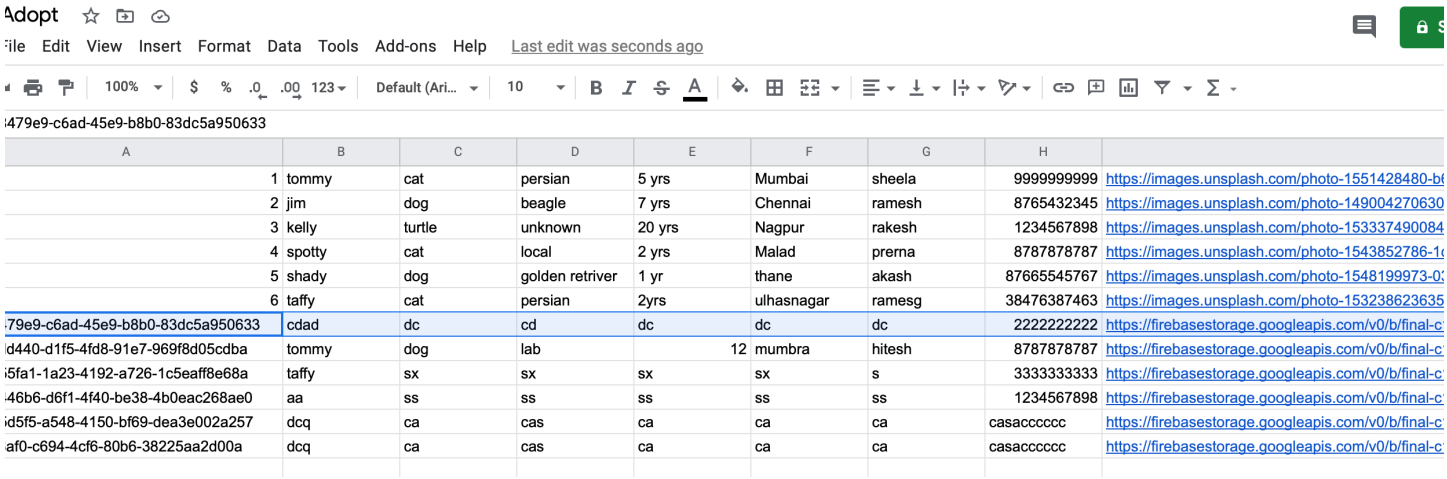


Snapshot of database in Cloud Firestore

- The part of app wherein pets are put up for adoption and the NGO's, rescuers can request for a donation amount is implemented using google sheets as backend database for ease of access and can be used by a person from non technical background.

- In order to accomplish this, a script is being run for the google sheet which in then deployed as web api, which transmits information to and fro (app to sheet and vice-a-versa) uses json object for communication.

- As we cannot store images in google sheets, they are being uploaded to firebase storage, and their respective links are fetched and stored in the sheet.



Adopt

File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

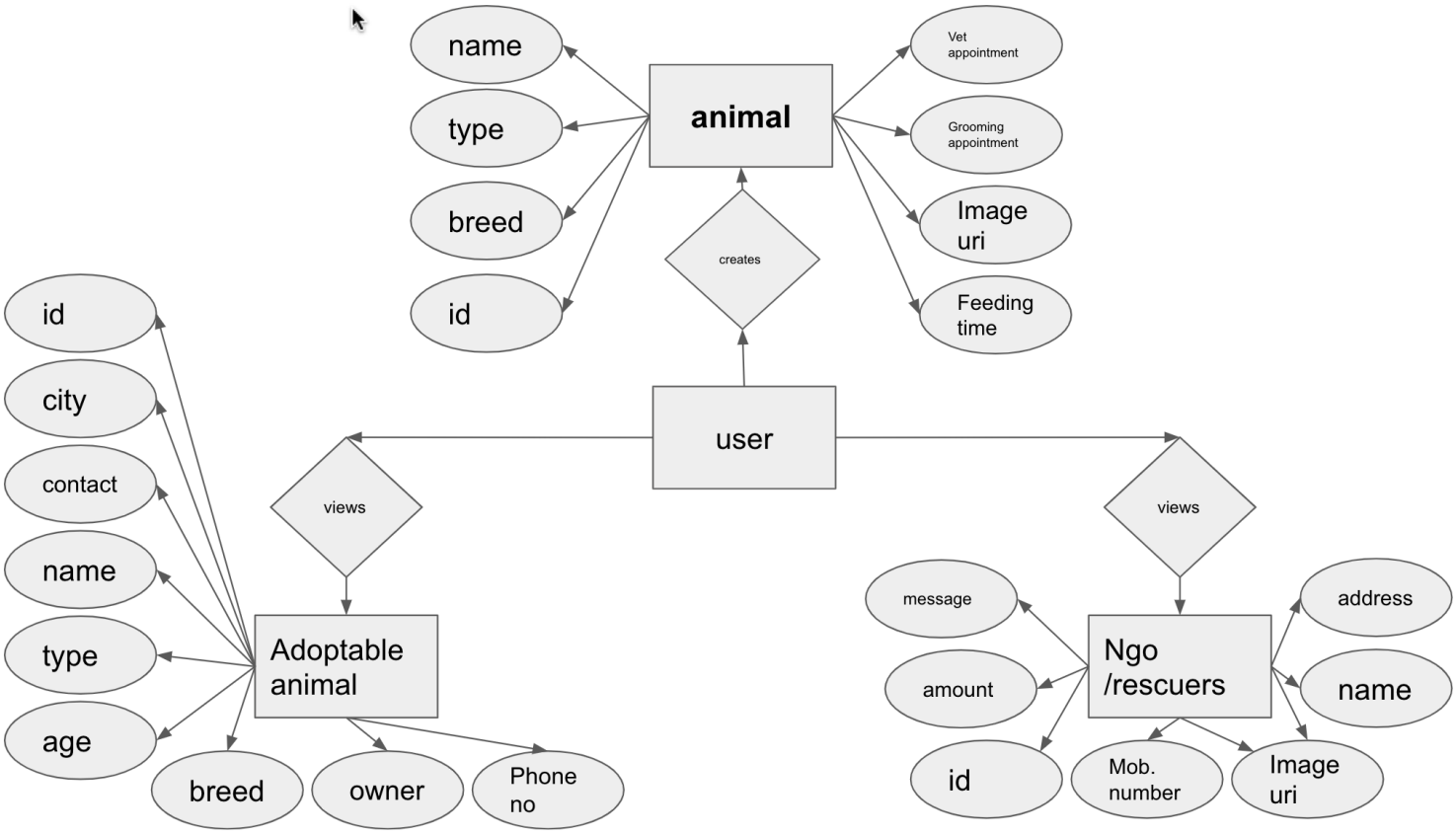
479e9-c6ad-45e9-b8b0-83dc5a950633

A	B	C	D	E	F	G	H	
1	tommy	cat	persian	5 yrs	Mumbai	sheela	9999999999	https://images.unsplash.com/photo-1551428480-b6
2	jim	dog	beagle	7 yrs	Chennai	ramesh	8765432345	https://images.unsplash.com/photo-149004270630
3	kelly	turtle	unknown	20 yrs	Nagpur	rakesh	1234567898	https://images.unsplash.com/photo-153337490084
4	spotty	cat	local	2 yrs	Malad	prerna	8787878787	https://images.unsplash.com/photo-1543852786-1
5	shady	dog	golden retriver	1 yr	thane	akash	87665545767	https://images.unsplash.com/photo-1548199973-03
6	taffy	cat	persian	2yrs	ulhasnagar	ramesg	38476387463	https://images.unsplash.com/photo-153238623635
79e9-c6ad-45e9-b8b0-83dc5a950633	cdad	dc	cd	dc	dc	dc	2222222222	https://firebasestorage.googleapis.com/v0/b/final-c
d440-d1f5-4fd8-91e7-969f8d05cdba	tommy	dog	lab	12	mumbra	hitesh	8787878787	https://firebasestorage.googleapis.com/v0/b/final-c
5fa1-1a23-4192-a726-1c5eaff8e68a	taffy	sx	sx	sx	sx	s	3333333333	https://firebasestorage.googleapis.com/v0/b/final-c
46b6-d6f1-4f40-be38-4b0eac268ae0	aa	ss	ss	ss	ss	ss	1234567898	https://firebasestorage.googleapis.com/v0/b/final-c
d5f5-a548-4150-bf69-dea3e002a257	dcq	ca	cas	ca	ca	ca	casacccccc	https://firebasestorage.googleapis.com/v0/b/final-c
af0-c694-4cf6-80b6-38225aa2d00a	dcq	ca	cas	ca	ca	ca	casacccccc	https://firebasestorage.googleapis.com/v0/b/final-c

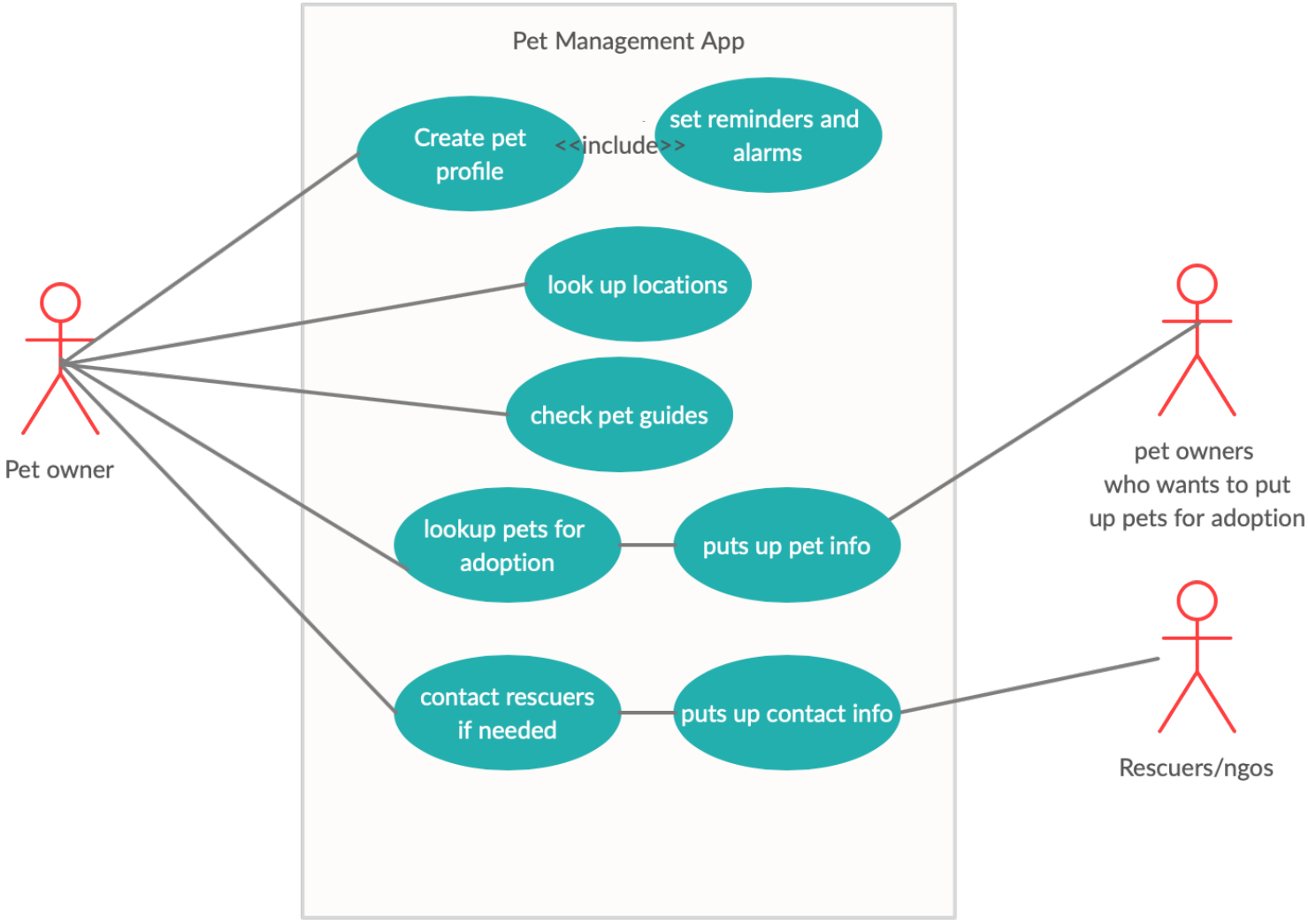
Snapshot of GoogleSheet storing pets to be adopted

DESIGN DETAILS

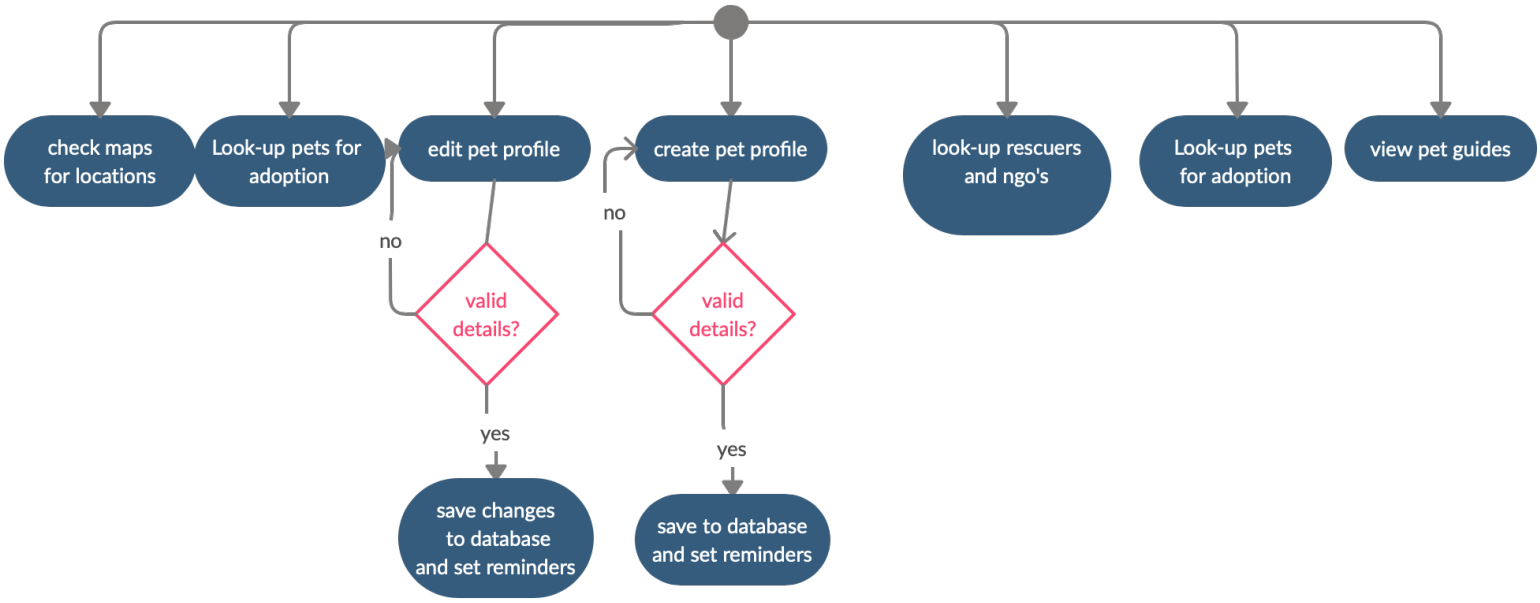
- **ENTITY RELATIONSHIP DIAGRAM**



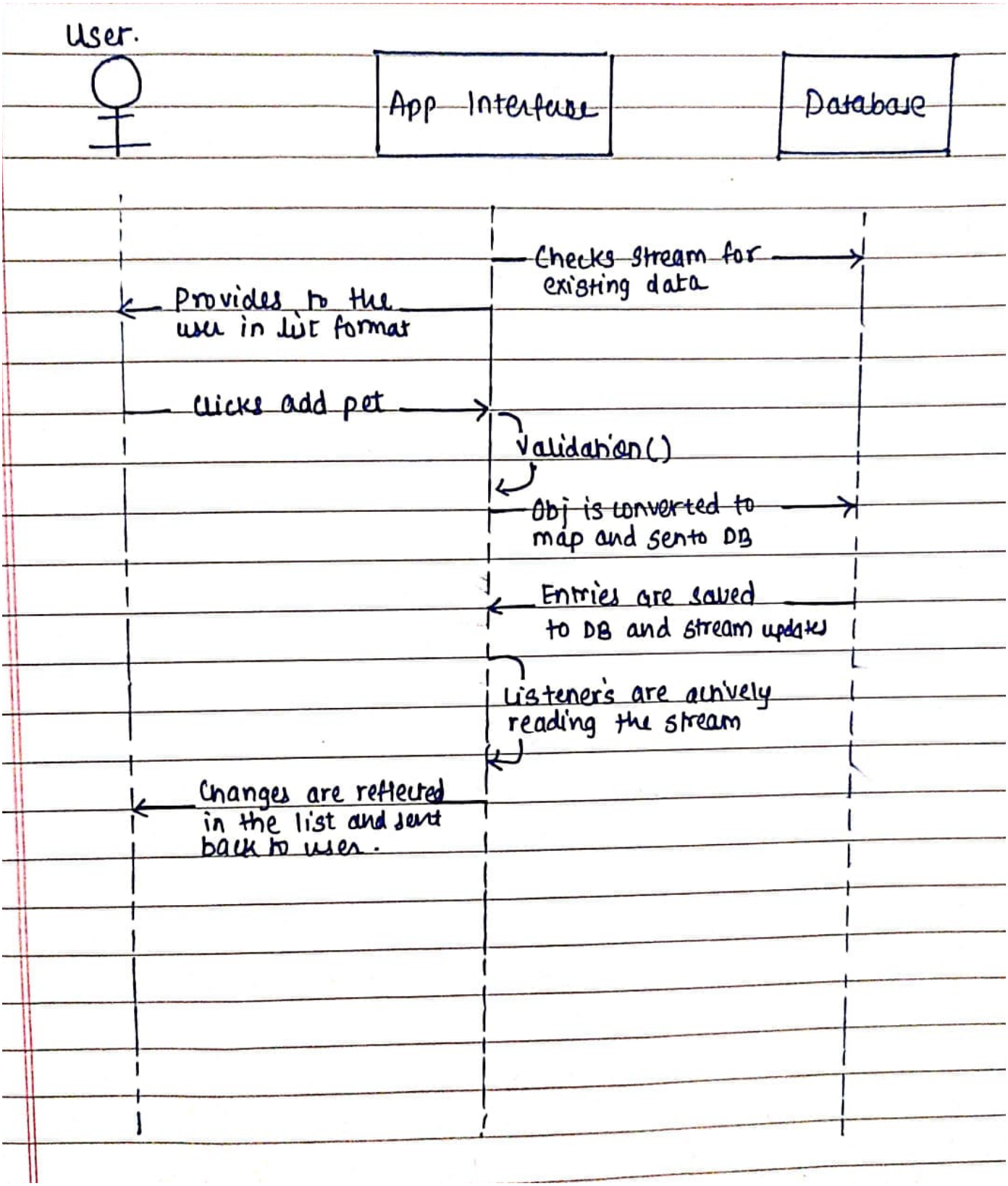
USE CASE DIAGRAM



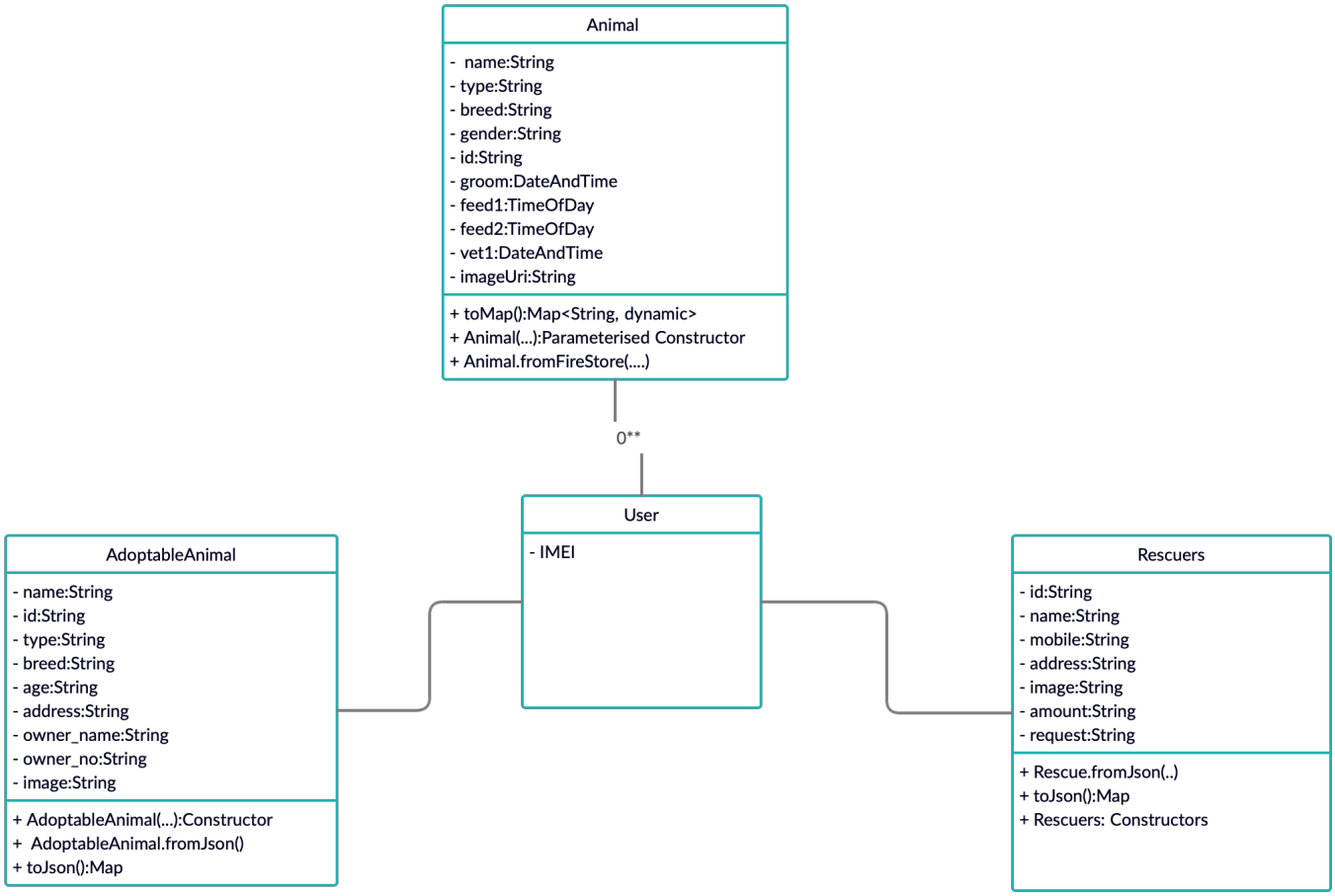
ACTIVITY DIAGRAM



● SEQUENCE DIAGRAM

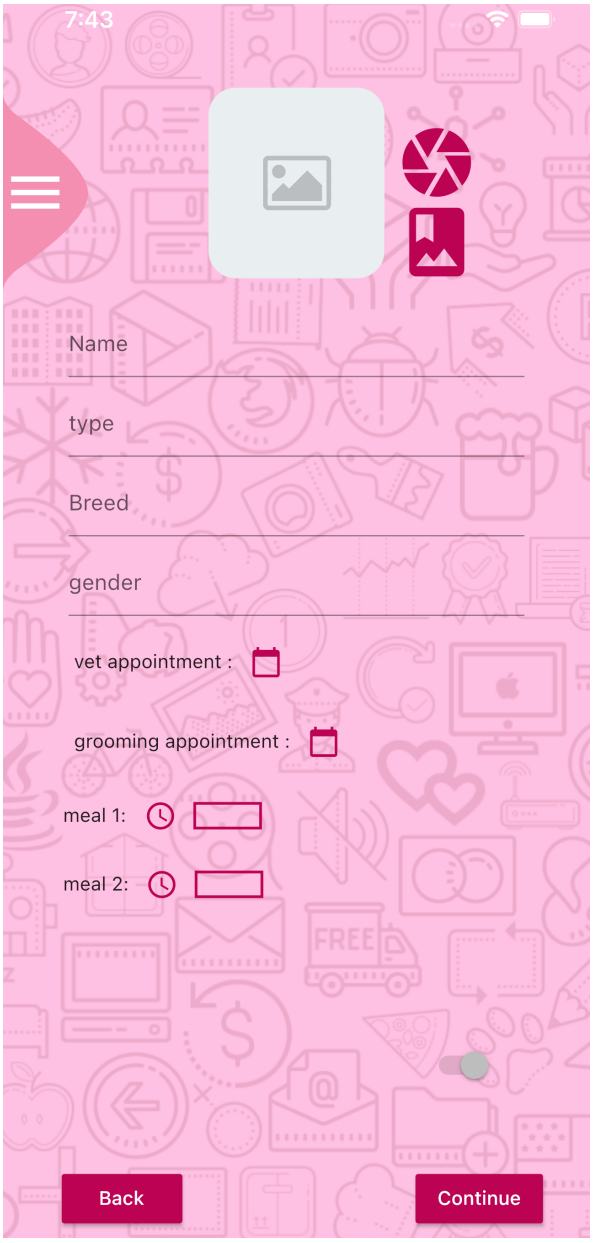


CLASS DIAGRAM

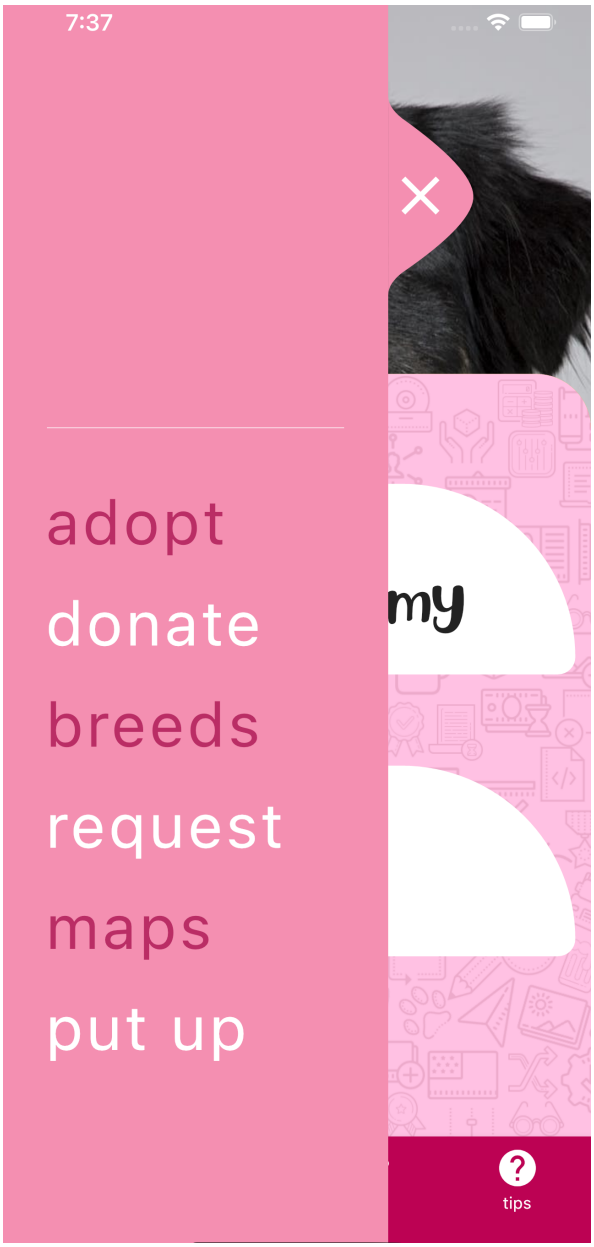


SCREENSHOTS

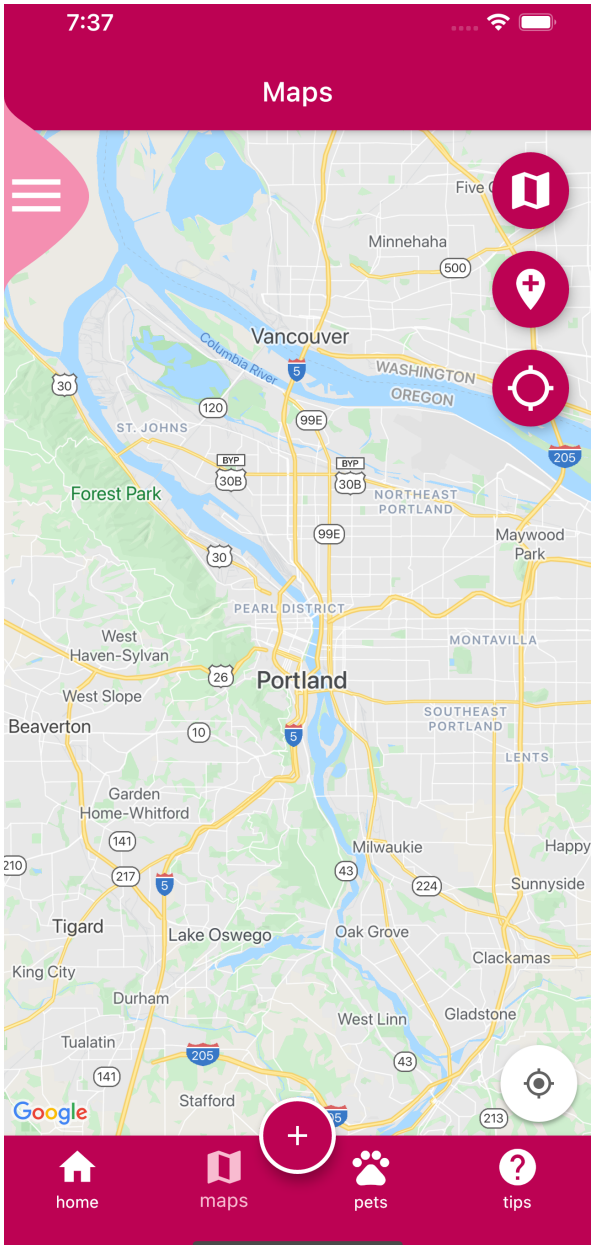
HOMESCREEN



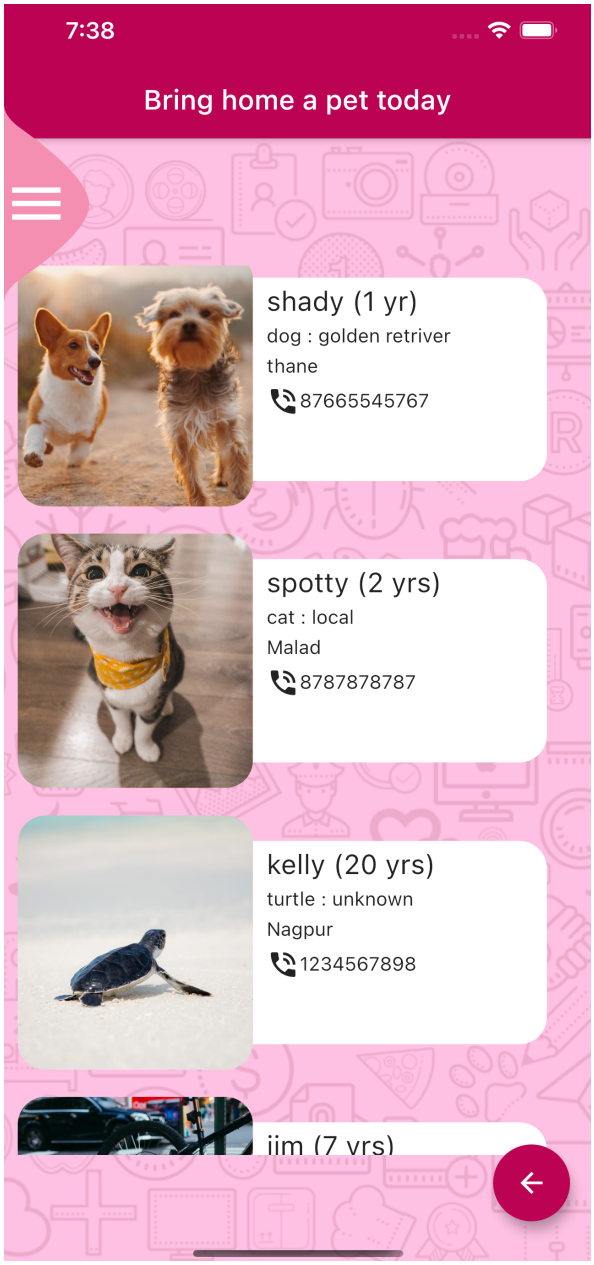
ADD A PET



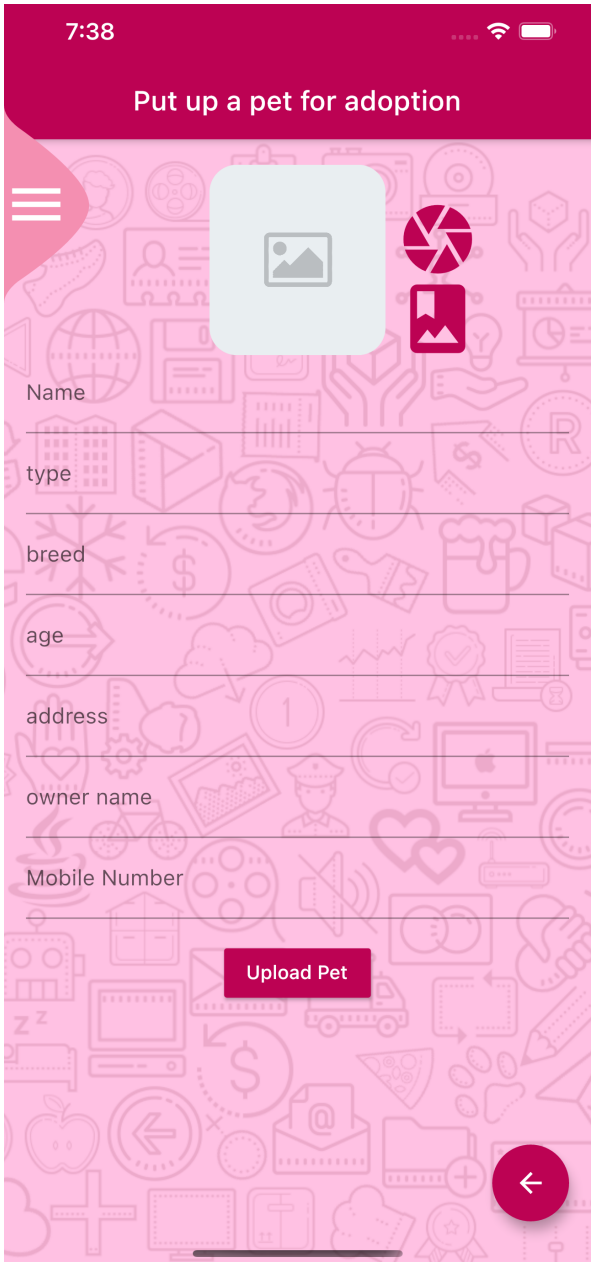
SIDEBAR



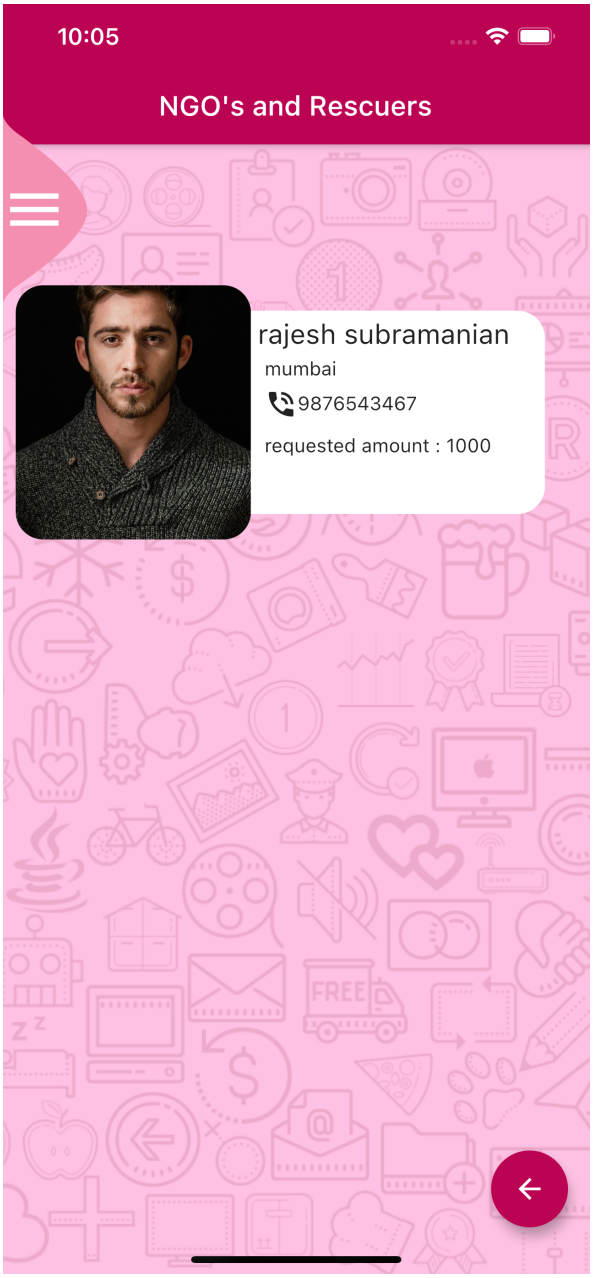
GOOGLE MAPS API



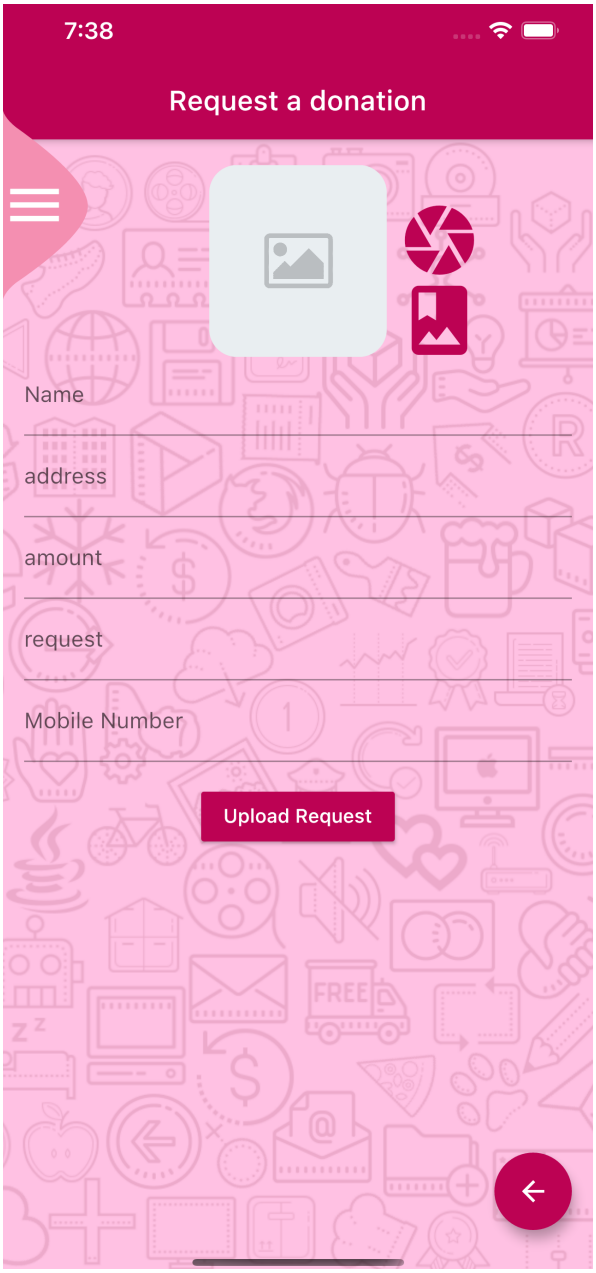
ADOPTION PAGE



PUT UP PETS FOR ADOPTION



NGO'S AND RESCUERS



REQUEST A DONATION

CODE

NAVIGATION_BLOC.DART

```
import 'package:Final/Pages/Adopt.dart';
import 'package:Final/Pages/Breeds.dart';
import 'package:Final/Pages/Ask_Donate.dart';
import 'package:Final/Pages/Rescuers.dart';
import 'package:Final/Pages/add-adopt_pet.dart';
import 'package:Final/Pages/homepage.dart';
import 'package:Final/Pages/Maps.dart';
import 'package:Final/Pages/add_pet.dart';

import 'package:bloc/bloc.dart';

enum NavigationEvents {
  AddPetClickedEvent,
  AboutUsClickedEvent,
  AdoptClickedEvent,
  BreedsClickedEvent,
  DonateClickedEvent,
  HelpClickedEvent,
  RescuersClickedEvent,
  ShopClickedEvent,
  HomePageClickedEvent,
}

abstract class NavigationStates {}

class NavigationBloc extends Bloc<NavigationEvents, NavigationStates> {
```

```
  @override

  NavigationStates get initialState => HomePage();

  @override
  Stream<NavigationStates> mapEventToState(NavigationEvents event) async*
{
    switch (event) {
      case NavigationEvents.HomePageClickedEvent:
        yield HomePage();
        break;
      case NavigationEvents.AboutUsClickedEvent:
        yield AboutUs();
        break;
      case NavigationEvents.AdoptClickedEvent:
        yield Adopt();
        break;
      case NavigationEvents.BreedsClickedEvent:
        yield Breeds();
        break;
      case NavigationEvents.DonateClickedEvent:
        yield Donate();
        break;
      case NavigationEvents.HelpClickedEvent:
        yield Help();
        break;
      case NavigationEvents.RescuersClickedEvent:
        yield Rescuers();
        break;

      case NavigationEvents.AddPetClickedEvent:
        yield AddPet();
        break;
```

```
    }

  }

}
```

ADOPT_PET.DART

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

class AdoptableAnimal {
  String id;
  String name;
  String type;
  String breed;
  String age;
  String address;
  String owner_name;
  String owner_no;
  String image;

  AdoptableAnimal(
      {@required this.id,
      @required this.name,
      @required this.type,
      @required this.breed,
      @required this.age,
      @required this.address,
      @required this.owner_name,
      @required this.owner_no,
      @required this.image});

  factory AdoptableAnimal.fromJson(dynamic json) {
    return AdoptableAnimal(
      id:"${json['id']}",
      name:"${json['name']}",
```

```
      type:"${json['type']}",
      breed:"${json['breed']}",
      age:"${json['age']}",
     address: "${json['address']}",
      owner_name:"${json['owner_name']}",
      owner_no:"${json['owner_no']}",
      image:"${json['image']}",
    );
  }

  // Method to make GET parameters.
  Map toJson() => {
        'id': id,
        'name': name,
        'type': type,
        'breed': breed,
        'age': age,
        'address': address,
        'owner_name': owner_name,
        'owner_no': owner_no,
        'image': image
      };
}
```

animal.dart

```
import 'dart:io';

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

class Animal {
  String name;
  String type;
  String breed;
  String gender;
  String id;
  String groom;
```

```
String feed1;
String feed2;
String vet1;
String image;

Animal({
  this.name,
  this.type,
  this.breed,
  this.gender,
  this.id,
  this.groom,
  this.feed1,
  this.feed2,
  this.vet1,
  this.image,
});

Map<String, dynamic> toMap() {
  return {
    "id": id,
    "name": name,
    "type": type,
    "breed": breed,
    "gender": gender,
    "vet1": vet1,
    "groom": groom,
    "feed1": feed1,
    "feed2": feed2,
    "image":image,
  };
}

Animal.fromFireStore(Map<String, dynamic> firestore)
    : id = firestore['id'],
      name = firestore['name'],
      type = firestore['type'],
      breed = firestore['breed'],
```

```
        gender = firestore['gender'],
        vet1=firestore['vet1'],
        groom=firestore['groom'],
       feed1=firestore['feed1'],
       feed2=firestore['feed2'],
       image=firestore['image'];
}
```

RESCUERS.DART

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

class Rescue {
  String id;
  String name;
  String mobile;
  String address;
  String image;
  String amount;
  String request;

  Rescue({
    @required this.id,
    @required this.name,
    @required this.mobile,
    @required this.address,
    @required this.image,
    @required this.amount,
    @required this.request,
  });

  factory Rescue.fromJson(dynamic json) {
    return Rescue(
      id:"${json['id']}",
      name:"${json['name']}",
      mobile:"${json['mobile']}",
      address:"${json['address']}",
      image:"${json['image']}",
```

```
      amount:"${json['amount']}",
      request:"${json['request']}",
    );
  }

  // Method to make GET parameters.
  Map toJson() => {
        'id': id,
        'name': name,
        'mobile': mobile,
        'address': address,
        'image': image,
        'amount': amount,
        'request': request,
      };
}
```

animalList.dart

```
import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:Final/Entitites/Animal.dart';
import 'package:Final/Pages/add_pet.dart';
import 'package:provider/provider.dart';

class AnimalList extends StatefulWidget {
  const AnimalList({Key key}) : super(key: key);
  @override
  _AnimalListState createState() => _AnimalListState();
}

class _AnimalListState extends State<AnimalList> {
  /*static MemoryImage imageFromBase64String(String base64String) {
    return MemoryImage(base64Decode(base64String));
  }*/
```

```
  @override
  Widget build(BuildContext context) {
    final animals = Provider.of<List<Animal>>(context);

    return Container(
      height: MediaQuery.of(context).size.height * 0.6,
      width: MediaQuery.of(context).size.width,
      child: (animals != null)
          ? ListView.builder(
              itemCount: animals.length,
              itemBuilder: (context, index) {
                return Padding(
                  padding: EdgeInsets.all(10),
                  child: GestureDetector(
                    onTap: () {
                      Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => AddPet(animals[index])));
                    },
                    child: Row(
                      children: [
                        ClipRRect(
                          borderRadius:
BorderRadius.all(Radius.circular(20)),
                          child: Image.network(
                            animals[index].image,
                            height: MediaQuery.of(context).size.height *
0.2,
                            width: MediaQuery.of(context).size.width *
0.40,
                            fit: BoxFit.cover,
                          ),
                        ),
                        Container(
                          width: MediaQuery.of(context).size.width *
0.55,
                          height: MediaQuery.of(context).size.height *
0.15,
                          decoration: BoxDecoration(
```

```
                                color: Colors.white,
                                borderRadius: BorderRadius.only(
                                    bottomLeft: Radius.circular(0),
                                    bottomRight: Radius.circular(15),
                                    topLeft: Radius.circular(0),
                                    topRight: Radius.circular(150))),
                            child: Align(alignment: Alignment.bottomLeft,
                              child: Padding(padding: EdgeInsets.all(20),
                                                                child: Text(
                                    animals[index].name,
                                    style: TextStyle(fontSize:
40,fontFamily:'blue',),
                                ),
                              ),
                            ),
                          )
                        ],
                      ),
                    ),
                  );
                },
              )
            : Center(child: CircularProgressIndicator()),
    );
  }
}
```

NOTIFICATIONS_PLUGIN.DART

```
import 'package:Final/Entitites/Animal.dart';
import 'package:flutter/material.dart';
import 'package:flutter_local_notifications/
flutter_local_notifications.dart';
import 'package:path_provider/path_provider.dart';
import 'dart:io' show File, Platform;
import 'package:http/http.dart' as http;
```

```dart
import 'package:rxdart/subjects.dart';

class NotificationPlugin {
  //
  FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin;
  final BehaviorSubject<ReceivedNotification>
      didReceivedLocalNotificationSubject =
      BehaviorSubject<ReceivedNotification>();
  var initializationSettings;

  NotificationPlugin._() {
    init();
  }

  init() async {
    flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
    if (Platform.isIOS) {
      _requestIOSPermission();
    }
    initializePlatformSpecifics();
  }

  initializePlatformSpecifics() {
    var initializationSettingsAndroid =
        AndroidInitializationSettings('iicon');
    var initializationSettingsIOS = IOSInitializationSettings(
      requestAlertPermission: true,
      requestBadgePermission: true,
      requestSoundPermission: false,
      onDidReceiveLocalNotification: (id, title, body, payload) async {
        ReceivedNotification receivedNotification = ReceivedNotification(
            id: id, title: title, body: body, payload: payload);
        didReceivedLocalNotificationSubject.add(receivedNotification);
      },
    );

    initializationSettings = InitializationSettings(android:
        initializationSettingsAndroid, iOS:initializationSettingsIOS);
```

```
  }

  _requestIOSPermission() {
    flutterLocalNotificationsPlugin
        .resolvePlatformSpecificImplementation<
            IOSFlutterLocalNotificationsPlugin>()
        .requestPermissions(
          alert: false,
          badge: true,
          sound: true,
        );
  }

  setListenerForLowerVersions(Function onNotificationInLowerVersions) {
    didReceivedLocalNotificationSubject.listen((receivedNotification) {
      onNotificationInLowerVersions(receivedNotification);
    });
  }

  setOnNotificationClick(Function onNotificationClick) async {
    await
flutterLocalNotificationsPlugin.initialize(initializationSettings,
        onSelectNotification: (String payload) async {
      onNotificationClick(payload);
    });
  }

  Future<void> showNotification() async {
    var androidChannelSpecifics = AndroidNotificationDetails(
      'CHANNEL_ID',
      'CHANNEL_NAME',
      "CHANNEL_DESCRIPTION",
      importance: Importance.max,
      priority: Priority.high,
      playSound: true,
      timeoutAfter: 5000,
      styleInformation: DefaultStyleInformation(true, true),
    );
```

```
    var iosChannelSpecifics = IOSNotificationDetails();
    var platformChannelSpecifics =
        NotificationDetails(android:androidChannelSpecifics,iOS:
iosChannelSpecifics);
    await flutterLocalNotificationsPlugin.show(
      0,
      'New Pet Added',
      '', //null
      platformChannelSpecifics,
      payload: 'New Payload',
    );
  }

  Future<void> showDailyAtTime(TimeOfDay timee,String title) async {
    var time = Time(timee.hour,timee.minute,0);
    var androidChannelSpecifics = AndroidNotificationDetails(
      'CHANNEL_ID 4',
      'CHANNEL_NAME 4',
      "CHANNEL_DESCRIPTION 4",
      importance: Importance.max,
      priority: Priority.high,
    );
    var iosChannelSpecifics = IOSNotificationDetails();
    var platformChannelSpecifics =
        NotificationDetails(android:androidChannelSpecifics,
iOS:iosChannelSpecifics);
    await flutterLocalNotificationsPlugin.showDailyAtTime(
      0,
      title,
      'Test Body', //null
      time,
      platformChannelSpecifics,
      payload: 'Test Payload',
    );
  }

  Future<void> showWeeklyAtDayTime() async {
    var time = Time(21, 5, 0);
```

```
    var androidChannelSpecifics = AndroidNotificationDetails(
      'CHANNEL_ID 5',
      'CHANNEL_NAME 5',
      "CHANNEL_DESCRIPTION 5",
      importance: Importance.max,
      priority: Priority.high,
    );
    var iosChannelSpecifics = IOSNotificationDetails();
    var platformChannelSpecifics =
        NotificationDetails(android:androidChannelSpecifics,
iOS:iosChannelSpecifics);
    await flutterLocalNotificationsPlugin.showWeeklyAtDayAndTime(
      0,
      'Test Title at ${time.hour}:${time.minute}.${time.second}',
      'Test Body', //null
      Day.saturday,
      time,
      platformChannelSpecifics,
      payload: 'Test Payload',
    );
  }

  Future<void> repeatNotification() async {
    var androidChannelSpecifics = AndroidNotificationDetails(
      'CHANNEL_ID 3',
      'CHANNEL_NAME 3',
      "CHANNEL_DESCRIPTION 3",
      importance: Importance.max,
      priority: Priority.high,
      styleInformation: DefaultStyleInformation(true, true),
    );
    var iosChannelSpecifics = IOSNotificationDetails();
    var platformChannelSpecifics =
        NotificationDetails(android:androidChannelSpecifics,iOS:
iosChannelSpecifics);
    await flutterLocalNotificationsPlugin.periodicallyShow(
      0,
      'Repeating Test Title',
```

```
    'Repeating Test Body',
    RepeatInterval.everyMinute,
    platformChannelSpecifics,
    payload: 'Test Payload',
  );
}

Future<void> scheduleNotification( DateTime date,String title) async {

  var scheduleNotificationDateTime = date;
  var androidChannelSpecifics = AndroidNotificationDetails(
    'CHANNEL_ID 1',
    'CHANNEL_NAME 1',
    "CHANNEL_DESCRIPTION 1",
    icon: 'iicon',

    largeIcon: DrawableResourceAndroidBitmap('iicon'),
    enableLights: true,
    color: const Color.fromARGB(255, 255, 0, 0),
    ledColor: const Color.fromARGB(255, 255, 0, 0),
    ledOnMs: 1000,
    ledOffMs: 500,
    importance: Importance.max,
    priority: Priority.high,
    playSound: true,
    timeoutAfter: 5000,
    styleInformation: DefaultStyleInformation(true, true),
  );
  var iosChannelSpecifics = IOSNotificationDetails(
    sound: 'my_sound.aiff',
  );
  var platformChannelSpecifics = NotificationDetails(android:
    androidChannelSpecifics,iOS:
    iosChannelSpecifics,
  );
  await flutterLocalNotificationsPlugin.schedule(
    0,
    title,
```

```
      '',
      scheduleNotificationDateTime,
      platformChannelSpecifics,
      payload: 'Test Payload',
    );
  }

  Future<void> showNotificationWithAttachment() async {
    var attachmentPicturePath = await _downloadAndSaveFile(
        'https://via.placeholder.com/800x200', 'attachment_img.jpg');
    var iOSPlatformSpecifics = IOSNotificationDetails(
      attachments: [IOSNotificationAttachment(attachmentPicturePath)],
    );
    var bigPictureStyleInformation = BigPictureStyleInformation(
      FilePathAndroidBitmap(attachmentPicturePath),
      contentTitle: '<b>Attached Image</b>',
      htmlFormatContentTitle: true,
      summaryText: 'Test Image',
      htmlFormatSummaryText: true,
    );
    var androidChannelSpecifics = AndroidNotificationDetails(
      'CHANNEL ID 2',
      'CHANNEL NAME 2',
      'CHANNEL DESCRIPTION 2',
      importance: Importance.high,
      priority: Priority.max,
      styleInformation: bigPictureStyleInformation,
    );
    var notificationDetails =
        NotificationDetails(android:androidChannelSpecifics,iOS:
iOSPlatformSpecifics);
    await flutterLocalNotificationsPlugin.show(
      0,
      'Title with attachment',
      'Body with Attachment',
      notificationDetails,
    );
  }
```

```
  _downloadAndSaveFile(String url, String fileName) async {
    var directory = await getApplicationDocumentsDirectory();
    var filePath = '${directory.path}/$fileName';
    var response = await http.get(url);
    var file = File(filePath);
    await file.writeAsBytes(response.bodyBytes);
    return filePath;
  }

  Future<int> getPendingNotificationCount() async {
    List<PendingNotificationRequest> p =
        await
flutterLocalNotificationsPlugin.pendingNotificationRequests();
    return p.length;
  }

  Future<void> cancelNotification() async {
    await flutterLocalNotificationsPlugin.cancel(0);
  }

  Future<void> cancelAllNotification() async {
    await flutterLocalNotificationsPlugin.cancelAll();
  }
}

NotificationPlugin notificationPlugin = NotificationPlugin._();

class ReceivedNotification {
  final int id;
  final String title;
  final String body;
  final String payload;

  ReceivedNotification({
    @required this.id,
    @required this.title,
    @required this.body,
```

```
    @required this.payload,
  });
}
```

add_pet.dart

```
import 'dart:convert';
import 'dart:io';

import 'package:Final/Entitites/Animal.dart';
import 'package:Final/providers/animal_provider.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
import 'package:intl/intl.dart';
import 'package:path/path.dart';
import 'package:path_provider/path_provider.dart';
import 'package:provider/provider.dart';
import 'package:Final/notifications/notification_plugins.dart';

class AddPet extends StatefulWidget with NavigationStates {
  final Animal animal;
  String id;
  AddPet([this.animal]);
  @override
  _AddPetState createState() => _AddPetState();
}

class _AddPetState extends State<AddPet> {
  DateTime vet = null;
  DateTime groom;
  TimeOfDay feed1;
  TimeOfDay feed2;
  String feed1_time = "";
```

```
  String feed2_time = "";
  String vet_date = "";
  String groom_date = "";
  bool isSwitched1 = true;
  String image;
  String id;

  File _selectedFile;

  final picker = ImagePicker();

  static var httpClient = new HttpClient();
Future<File> _downloadFile(String url, String filename) async {
  var request = await httpClient.getUrl(Uri.parse(url));
  var response = await request.close();
  var bytes = await consolidateHttpClientResponseBytes(response);
  String dir = (await getApplicationDocumentsDirectory()).path;
  File file = new File('$dir/$filename');
  await file.writeAsBytes(bytes);
  return file;
}

  Future<String> uploadImageToFirebase(BuildContext context) async {
    String fileName = basename(_selectedFile.path);
    Reference firebaseStorageRef =
        FirebaseStorage.instance.ref().child('photos/$fileName');
    UploadTask uploadTask = firebaseStorageRef.putFile(_selectedFile);
    final ref = FirebaseStorage.instance.ref().child('photos/$fileName');

    var image = await ref.getDownloadURL();
    return image;
  }

  onNotificationInLowerVersions(ReceivedNotification
receivedNotification) {}
  onNotificationClick(String payload) {}

  /*static String base64String(Uint8List data) {
```

```
  return base64Encode(data);
}*/

/*static MemoryImage imageFromBase64String(String base64String) {
  return MemoryImage(base64Decode(base64String));}*/

getMyImage(ImageSource source) async {
  final pickedFile = await picker.getImage(source: source);
  if (pickedFile != null) {
    File cropped = await ImageCropper.cropImage(
      sourcePath: pickedFile.path,
      aspectRatio: CropAspectRatio(ratioX: 1, ratioY: 1),
      compressQuality: 100,
      maxHeight: 700,
      maxWidth: 700,
      compressFormat: ImageCompressFormat.jpg,
    );
    this.setState(() {
      _selectedFile = cropped;
    });
  }
}

Widget getimageWidget() {
  if (_selectedFile != null) {
    return ClipRRect(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      child: Image.file(
        _selectedFile,
        height: MediaQuery.of(this.context).size.height * 0.15,
        width: MediaQuery.of(this.context).size.width * 0.3,
        fit: BoxFit.cover,
      ),
    );
  } else {
    return ClipRRect(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      child: Image.asset(
```

```
          "assets/images/dd.png",
          height: MediaQuery.of(this.context).size.height * 0.15,
          width: MediaQuery.of(this.context).size.width * 0.3,
          fit: BoxFit.cover,
        ),
      );
    }
  }

final nameController = TextEditingController();
final typeController = TextEditingController();
final breedController = TextEditingController();
final genderController = TextEditingController();

void dispose() {
  nameController.dispose();
  typeController.dispose();
  breedController.dispose();
  genderController.dispose();
  super.dispose();
}

@override
void initState()  {
  if (widget.animal == null) {
    nameController.text = '';
    typeController.text = '';
    breedController.text = '';
    genderController.text = '';
    new Future.delayed(Duration.zero, () {
      final animalProvider =
          Provider.of<AnimalProvider>(this.context, listen: false);
      animalProvider.loadValues(Animal());
    });
  } else {
    nameController.text = widget.animal.name;
    typeController.text = widget.animal.type;
    breedController.text = widget.animal.breed;
```

```
      genderController.text = widget.animal.gender;

      feed1_time = widget.animal.feed1.substring(10, 15);
      feed2_time = widget.animal.feed2.substring(10, 15);
      vet_date = DateFormat.yMEd().format(DateTime(
          int.parse(widget.animal.vet1.substring(0, 4)),
          int.parse(widget.animal.vet1.substring(8, 10)),
          int.parse(widget.animal.vet1.substring(5, 7))));
      groom_date = DateFormat.yMEd().format(DateTime(
          int.parse(widget.animal.groom.substring(0, 4)),
          int.parse(widget.animal.groom.substring(8, 10)),
          int.parse(widget.animal.groom.substring(5, 7))));

      //MemoryImage image = imageFromBase64String(widget.animal.image);

     // _selectedFile =await _downloadFile(widget.animal.image,
widget.animal.id);
      new Future.delayed(Duration.zero, () {
        final animalProvider =
            Provider.of<AnimalProvider>(this.context, listen: false);
        animalProvider.loadValues(widget.animal);
      });
    }
    super.initState();
    vet = DateTime.now();
    groom = DateTime.now();
    feed1 = TimeOfDay.now();
    feed2 = TimeOfDay.now();

    notificationPlugin
        .setListenerForLowerVersions(onNotificationInLowerVersions);
    notificationPlugin.setOnNotificationClick(onNotificationClick);
  }

  @override
  Widget build(BuildContext context) {
    final animalProvider = Provider.of<AnimalProvider>(context);
```

```
    return Scaffold(
      body: SingleChildScrollView(
        child: Stack(
          children: [
            Image(
              image: AssetImage('assets/images/bg.png'),
              width: MediaQuery.of(context).size.width,
              height: MediaQuery.of(context).size.height,
              fit: BoxFit.cover,
            ),
            Positioned(
                top: MediaQuery.of(context).size.height * 0.08,
                left: MediaQuery.of(context).size.width * 0.35,
                child: Row(
                  children: [
                    getimageWidget(),
                    Column(
                      children: [
                        IconButton(
                            icon: Icon(Icons.camera,
                                color: Color(0xffBC0253),
                                size:
                                    MediaQuery.of(context).size.height *
0.065),
                            onPressed: () {
                              getMyImage(ImageSource.camera);
                            }),
                        SizedBox(
                          height: 10,
                        ),
                        IconButton(
                            icon: Icon(Icons.photo_album,
                                color: Color(0xffBC0253),
                                size:
                                    MediaQuery.of(context).size.height *
0.065),
                            onPressed: () {
                              getMyImage(ImageSource.gallery);
```

```
                    })
                ],
              )
            ],
          )),
      Positioned(
        top: MediaQuery.of(context).size.height * 0.25,
        left: MediaQuery.of(context).size.width * 0.1,
        child: Column(
          children: [
            Container(
              padding: EdgeInsets.all(5),
              width: MediaQuery.of(context).size.width * 0.8,
              child: TextField(
                controller: nameController,
                onChanged: (value) {
                  animalProvider.chnageName(value);
                },
                decoration: InputDecoration(hintText: 'Name'),
              ),
            ),
            Container(
              padding: EdgeInsets.all(5),
              width: MediaQuery.of(context).size.width * 0.8,
              child: TextField(
                controller: typeController,
                onChanged: (value) {
                  animalProvider.chnageType(value);
                },
                decoration: InputDecoration(hintText: 'type'),
              ),
            ),
            Container(
              padding: EdgeInsets.all(5),
              width: MediaQuery.of(context).size.width * 0.8,
              child: TextField(
                controller: breedController,
                onChanged: (value) {
```

```
                    animalProvider.chnageBreed(value);
                  },
                  decoration: InputDecoration(hintText: 'Breed'),
                ),
              ),
              Container(
                padding: EdgeInsets.all(5),
                width: MediaQuery.of(context).size.width * 0.8,
                child: TextField(
                  controller: genderController,
                  onChanged: (value) {
                    animalProvider.chnageGender(value);
                  },
                  decoration: InputDecoration(hintText: 'gender'),
                ),
              ),
              Container(
                padding: EdgeInsets.all(5),
                width: MediaQuery.of(context).size.width * 0.8,
                child: Row(
                  children: [
                    Text(" vet appointment :"),
                    IconButton(
                      onPressed: () async {
                        DateTime date = await showDatePicker(
                            context: context,
                            initialDate: vet,
                            firstDate: DateTime(DateTime.now().year -
5),
                            lastDate: DateTime(DateTime.now().year +
5));
                        if (date != null) {
                          setState(() {
                            vet = date;
                            vet_date = DateFormat.yMEd().format(vet);
                          });
                        }
                      },
```

```
                        icon: Icon(
                          Icons.calendar_today,
                          color: Color(0xffBC0253),
                        ),
                      ),
                      Expanded(child: Text(vet_date))
                    ],
                  ),
                ),
                Container(
                  padding: EdgeInsets.all(5),
                  width: MediaQuery.of(context).size.width * 0.8,
                  child: Row(
                    children: [
                      Text(" grooming appointment :"),
                      IconButton(
                        onPressed: () async {
                          DateTime date = await showDatePicker(
                            context: context,
                            initialDate: groom,
                            firstDate: DateTime(DateTime.now().year -
5),
                            lastDate: DateTime(DateTime.now().year +
5),
                          );
                          if (date != null) {
                            setState(() {
                              groom = date;
                              groom_date =
DateFormat.yMEd().format(groom);
                            });
                          }
                        },
                        icon: Icon(
                          Icons.calendar_today,
                          color: Color(0xffBC0253),
                        ),
                      ),
```

```
        Expanded(child: Text(groom_date))
      ],
    ),
  ),
  Container(
    padding: EdgeInsets.all(1),
    width: MediaQuery.of(context).size.width * 0.8,
    child: Row(children: [
      Text('meal 1:'),
      IconButton(
          icon: Icon(
            Icons.access_time,
            color: Color(0xffBC0253),
          ),
          onPressed: () async {
            TimeOfDay t = await showTimePicker(
                context: context, initialTime: feed1);
            if (t != null) {
              setState(() {
                feed1 = t;
                feed1_time = t.hour.toString() +
                    ":" +
                    t.minute.toString();
              });
            }
          }),
      Container(
          height: 20,
          width: 50,
          decoration: BoxDecoration(
              border: Border.all(
            width: 2,
            color: Color(0xffBC0253),
          )),
          child: Center(child: Text("${feed1_time}"))),
    ]),
  ),
  Container(
```

```
            padding: EdgeInsets.all(1),
            width: MediaQuery.of(context).size.width * 0.8,
            child: Row(children: [
              Text('meal 2:'),
              IconButton(
                  icon: Icon(
                    Icons.access_time,
                    color: Color(0xffBC0253),
                  ),
                  onPressed: () async {
                    TimeOfDay t = await showTimePicker(
                        context: context, initialTime: feed2);
                    if (t != null) {
                      setState(() {
                        feed2 = t;
                        feed2_time = t.hour.toString() +
                            ":" +
                            t.minute.toString();
                      });
                    }
                  }),
              Container(
                  height: 20,
                  width: 50,
                  decoration: BoxDecoration(
                      border: Border.all(
                    width: 2,
                    color: Color(0xffBC0253),
                  )),
                  child: Center(child: Text("${feed2_time}"))),
            ]),
          ),
        ],
      ),
    ),
    (widget.animal != null)
        ? Positioned(
            bottom: MediaQuery.of(context).size.height * 0.02,
```

```
                right: MediaQuery.of(context).size.width * 0.375,
                child: RaisedButton(
                  onPressed: () {
                    animalProvider.removeAnimal(widget.animal.id);
                    if (widget.animal == null) {
                      BlocProvider.of<NavigationBloc>(context)
                          .add(NavigationEvents.HomePageClickedEvent)
;
                    } else {
                      Navigator.of(context).pop();
                    }
                  },
                  child: Text(
                    'Delete',
                    style: TextStyle(color: Colors.white),
                  ),
                  color: Color(0xffBC0253),
                ))
            : Container(),
        Positioned(
            bottom: MediaQuery.of(context).size.height * 0.02,
            left: MediaQuery.of(context).size.width * 0.1,
            child: RaisedButton(
              onPressed: () {
                if (widget.animal == null) {
                  BlocProvider.of<NavigationBloc>(context)
                      .add(NavigationEvents.HomePageClickedEvent);
                } else {
                  Navigator.of(context).pop();
                }
              },
              child: Text(
                'Back',
                style: TextStyle(color: Colors.white),
              ),
              color: Color(0xffBC0253),
            )),
        Positioned(
```

```
                bottom: MediaQuery.of(context).size.height * 0.02,
                right: MediaQuery.of(context).size.width * 0.08,
                child: RaisedButton(
                  onPressed: () async {
                    //image =
base64String(_selectedFile.readAsBytesSync());
                    String image = await uploadImageToFirebase(context);
                    await notificationPlugin.showNotification();
                    await notificationPlugin.showDailyAtTime(
                        feed1, "${nameController.text} is hungry");
                    await notificationPlugin.showDailyAtTime(
                        feed2, "${nameController.text} is hungry");
                    await notificationPlugin.scheduleNotification(groom,
                        "${nameController.text} has a vet grooming
today");

                    await notificationPlugin.scheduleNotification(vet,
                        "${nameController.text} has a vet appointment
today");
                    animalProvider.chnagevet(vet, groom, feed1, feed2,
image);
                    animalProvider.saveProduct();
                    if (widget.animal == null) {
                      BlocProvider.of<NavigationBloc>(context)
                          .add(NavigationEvents.HomePageClickedEvent);
                    } else {
                      Navigator.of(context).pop();
                    }
                  },
                  child: Text(
                    'Continue',
                    style: TextStyle(color: Colors.white),
                  ),
                  color: Color(0xffBC0253),
                )),
            Positioned(
              right: MediaQuery.of(context).size.width * 0.15,
              bottom: MediaQuery.of(context).size.height * 0.125,
```

```
              child: Switch(
                value: isSwitched1,
                onChanged: null,
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

ADD-ADOPT_PET.DART

```
import 'dart:io';

import 'package:Final/Entitites/Adopt_pet.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
import 'package:Final/providers/Adopt_provider.dart';
import 'package:path/path.dart';
import 'package:uuid/uuid.dart';

class AboutUs extends StatefulWidget with NavigationStates {
  @override
  _AboutUsState createState() => _AboutUsState();
}

class _AboutUsState extends State<AboutUs> {
  final _formKey = GlobalKey<FormState>();
  final _scaffoldKey = GlobalKey<ScaffoldState>();
  var uuid = Uuid();

  TextEditingController nameController = TextEditingController();
```

```
TextEditingController typeController = TextEditingController();
TextEditingController breedController = TextEditingController();
TextEditingController ageController = TextEditingController();
TextEditingController addressController = TextEditingController();
TextEditingController ownerController = TextEditingController();
TextEditingController mobileNoController = TextEditingController();

File _selectedFile;

final picker = ImagePicker();

getMyImage(ImageSource source) async {
  final pickedFile = await picker.getImage(source: source);
  if (pickedFile != null) {
    File cropped = await ImageCropper.cropImage(
      sourcePath: pickedFile.path,
      aspectRatio: CropAspectRatio(ratioX: 1, ratioY: 1),
      compressQuality: 100,
      maxHeight: 700,
      maxWidth: 700,
      compressFormat: ImageCompressFormat.jpg,
    );
    this.setState(() {
      _selectedFile = cropped;
    });
  }
}

Widget getimageWidget() {
  if (_selectedFile != null) {
    return ClipRRect(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      child: Image.file(
        _selectedFile,
        height: MediaQuery.of(this.context).size.height * 0.15,
        width: MediaQuery.of(this.context).size.width * 0.3,
        fit: BoxFit.cover,
      ),
```

```
      );
    } else {
      return ClipRRect(
        borderRadius: BorderRadius.all(Radius.circular(20)),
        child: Image.asset(
          "assets/images/dd.png",
          height: MediaQuery.of(this.context).size.height * 0.15,
          width: MediaQuery.of(this.context).size.width * 0.3,
          fit: BoxFit.cover,
        ),
      );
    }
  }

  Future<String> uploadImageToFirebase(BuildContext context) async {
    String fileName = basename(_selectedFile.path);
    Reference firebaseStorageRef =
        FirebaseStorage.instance.ref().child('adoption/$fileName');
    UploadTask uploadTask = firebaseStorageRef.putFile(_selectedFile);
    final ref = FirebaseStorage.instance.ref().child('adoption/
$fileName');

    var image = await ref.getDownloadURL();
    return image;
  }

  void _submitForm(String image) {
    if (_formKey.currentState.validate()) {
      AdoptableAnimal a = AdoptableAnimal(
          id:uuid.v4(),
          name:nameController.text,
          type:typeController.text,
          breed:breedController.text,
          age:ageController.text,
          address:addressController.text,
          owner_name:ownerController.text,
          owner_no:mobileNoController.text,
          image:image,);
```

```
      FormController formController = FormController();

      _showSnackbar("Submitting Feedback");

      // Submit 'feedbackForm' and save it in Google Sheets.
      formController.submitForm(a, (String response) {
        print("Response: $response");
        if (response == FormController.STATUS_SUCCESS) {
          // Feedback is saved succesfully in Google Sheets.
          _showSnackbar("Feedback Submitted");
        } else {
          // Error Occurred while saving data in Google Sheets.
          _showSnackbar("Error Occurred!");
        }
      });
    }
  }

  // Method to show snackbar with 'message'.
  _showSnackbar(String message) {
    final snackBar = SnackBar(content: Text(message));
    _scaffoldKey.currentState.showSnackBar(snackBar);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      resizeToAvoidBottomPadding: false,
      floatingActionButtonLocation:
FloatingActionButtonLocation.endFloat,
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.arrow_back),
        backgroundColor: Color(0xffBC0253),
        onPressed: () {
          BlocProvider.of<NavigationBloc>(context)
              .add(NavigationEvents.HomePageClickedEvent);
```

```
        },
      ),
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "Put up a pet for adoption",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(children: [
        Image(
          image: AssetImage('assets/images/bg.png'),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          fit: BoxFit.cover,
        ),
        Positioned(
            top: MediaQuery.of(context).size.height * 0.02,
            left: MediaQuery.of(context).size.width * 0.35,
            child: Row(
              children: [
                getimageWidget(),
                Column(
                  children: [
                    IconButton(
                        icon: Icon(Icons.camera,
                            color: Color(0xffBC0253),
                            size: MediaQuery.of(context).size.height *
0.065),
                        onPressed: () {
                          getMyImage(ImageSource.camera);
                        }),
                    SizedBox(
                      height: 10,
                    ),
                    IconButton(
                        icon: Icon(Icons.photo_album,
                            color: Color(0xffBC0253),
```

```
                            size: MediaQuery.of(context).size.height *
0.065),
                        onPressed: () {
                          getMyImage(ImageSource.gallery);
                        })
                  ],
                ),
              ],
            )),
        Positioned(
          top: MediaQuery.of(context).size.height * 0.15,
          width: MediaQuery.of(context).size.width,
          child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              children: <Widget>[
                Form(
                    key: _formKey,
                    child: Padding(
                      padding: EdgeInsets.all(16),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: <Widget>[
                          TextFormField(
                            controller: nameController,
                            validator: (value) {
                              if (value.isEmpty) {
                                return 'Enter Valid Name';
                              }
                              return null;
                            },
                            decoration: InputDecoration(labelText:
'Name'),
                          ),
                          TextFormField(
                            controller: typeController,
                            validator: (value) {
                              if (value.isEmpty) {
                                return 'Enter Valid type';
```

```
                    }
                    return null;
                  },
                  decoration: InputDecoration(labelText:
'type'),
                ),
                TextFormField(
                  controller: breedController,
                  validator: (value) {
                    if (value.isEmpty) {
                      return 'Enter Valid breed';
                    }
                    return null;
                  },
                  decoration: InputDecoration(labelText:
'breed'),
                ),
                TextFormField(
                  controller: ageController,
                  validator: (value) {
                    if (value.isEmpty) {
                      return 'Enter Valid age';
                    }
                    return null;
                  },
                  decoration: InputDecoration(labelText:
'age'),
                ),
                TextFormField(
                  controller: addressController,
                  validator: (value) {
                    if (value.isEmpty) {
                      return 'Enter Valid address';
                    }
                    return null;
                  },
                  decoration: InputDecoration(labelText:
'address'),
```

```
            ),
            TextFormField(
              controller: ownerController,
              validator: (value) {
                if (value.isEmpty) {
                  return 'Enter Valid owner name';
                }
                return null;
              },
              decoration:
                  InputDecoration(labelText: 'owner name'),
            ),
            TextFormField(
              controller: mobileNoController,
              validator: (value) {
                if (value.trim().length != 10) {
                  return 'Enter 10 Digit Mobile Number';
                }
                return null;
              },
              keyboardType: TextInputType.number,
              decoration: InputDecoration(
                labelText: 'Mobile Number',
              ),
            ),
          ],
        ),
      )),
RaisedButton(
  color: Color(0xffBC0253),
  textColor: Colors.white,
  onPressed: () async {
    String image = await uploadImageToFirebase(context);
    print(image);
    _submitForm(image);
  },
  child: Text('Upload Pet'),
),
```

```
                ]),
          )
        ]),
    );
  }
}
```

ADOPT.DART

```
import 'dart:ui';
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:Final/Entitites/Adopt_pet.dart';

class Adopt extends StatefulWidget with NavigationStates {
  @override
  _AdoptState createState() => _AdoptState();
}

class _AdoptState extends State<Adopt> {
  List<AdoptableAnimal> adoptList = List<AdoptableAnimal>();

  String url =
      "https://script.google.com/macros/s/
AKfycbwEl2uTHn67dlKDedEy87q_EWAt47kL4UaeAnSmQ7SmFuX0j58/exec";

  Future<List<AdoptableAnimal>> getFeedbackList() async {
    return await http.get(url).then((response) {
      var jsonFeedback = convert.jsonDecode(response.body) as List;
      return jsonFeedback
          .map((json) => AdoptableAnimal.fromJson(json))
          .toList();
    });
  }
```

```
  @override
  void initState() {
    super.initState();

    getFeedbackList().then((feedbackItems) {
      setState(() {
        this.adoptList = feedbackItems;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButtonLocation:
FloatingActionButtonLocation.endFloat,
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.arrow_back),
        backgroundColor: Color(0xffBC0253),
        onPressed: () {
          BlocProvider.of<NavigationBloc>(context)
              .add(NavigationEvents.HomePageClickedEvent);
        },
      ),
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "Bring home a pet today",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(
        children: [
          Image(
            image: AssetImage('assets/images/bg.png'),
            width: MediaQuery.of(context).size.width,
            height: MediaQuery.of(context).size.height,
            fit: BoxFit.cover,
```

```
          ),
          Positioned(
            top: MediaQuery.of(context).size.height * 0.1,
            child: Container(
              height: MediaQuery.of(context).size.height * 0.7,
              width: MediaQuery.of(context).size.width,
              child: ListView.builder(
                itemCount: adoptList.length,
                itemBuilder: (context, index) {
                  return Padding(
                    padding: EdgeInsets.all(10),
                    child: Row(
                      children: [
                        ClipRRect(
                          borderRadius: BorderRadius.circular(20),
                          child: Image.network(
                            "${adoptList[index].image}",
                            width: MediaQuery.of(context).size.width *
0.4,
                            height: MediaQuery.of(context).size.height *
0.2,
                            fit: BoxFit.cover,
                          ),
                        ),
                        Container(
                          height: MediaQuery.of(context).size.height *
0.16,
                          width: MediaQuery.of(context).size.width * 0.5,
                          decoration: BoxDecoration(
                            borderRadius: BorderRadius.only(
                                bottomRight: Radius.circular(20),
                                topRight: Radius.circular(20)),
                            color: Colors.white,
                          ),
                          child: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Padding(
```

```
                                padding: EdgeInsets.fromLTRB(10,5,5,0),
                                child: Text(
                                  '${adoptList[index].name}' +
                                      " " +
                                      "(" +
                                      "${adoptList[index].age}" +
                                      ")",
                                  style: TextStyle(fontSize: 20),
                                )),
                            Padding(
                                padding: EdgeInsets.fromLTRB(10, 5, 5,
0),
                                child: Text("$
{adoptList[index].type}"+" : "+'${adoptList[index].breed}')),
                            Padding(
                                padding: EdgeInsets.fromLTRB(10, 5, 5,
0),
                                child: Text('$
{adoptList[index].address}')),
                            Padding(padding: EdgeInsets.fromLTRB(10, 5,
5, 0),
                                                                child: Row(
                                children: [
                                  Icon(Icons.phone_in_talk),
                                  Text("${adoptList[index].owner_no}")
                                ],
                              ),
                            ),
                          ],
                        ),
                      )
                    ],
                  ),
                );
              },
            ),
          ),
        )
```

```
          ],
        ),
      );
    }
}
```

ASK_DONATE.DART

```
import 'dart:io';

import 'package:Final/Entitites/rescuers.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
import 'package:Final/providers/Rescue_provider.dart';
import 'package:path/path.dart';
import 'package:uuid/uuid.dart';

class Donate extends StatefulWidget with NavigationStates {
  @override
  _Donate createState() => _Donate();
}

class _Donate extends State<Donate> {
  final _formKey = GlobalKey<FormState>();
  final _scaffoldKey = GlobalKey<ScaffoldState>();
  var uuid = Uuid();

  TextEditingController nameController = TextEditingController();
  TextEditingController amountController = TextEditingController();
  TextEditingController requestController = TextEditingController();
  TextEditingController addressController = TextEditingController();
  TextEditingController mobileNoController = TextEditingController();

  File _selectedFile;
```

```
final picker = ImagePicker();

getMyImage(ImageSource source) async {
  final pickedFile = await picker.getImage(source: source);
  if (pickedFile != null) {
    File cropped = await ImageCropper.cropImage(
      sourcePath: pickedFile.path,
      aspectRatio: CropAspectRatio(ratioX: 1, ratioY: 1),
      compressQuality: 100,
      maxHeight: 700,
      maxWidth: 700,
      compressFormat: ImageCompressFormat.jpg,
    );
    this.setState(() {
      _selectedFile = cropped;
    });
  }
}

Widget getimageWidget() {
  if (_selectedFile != null) {
    return ClipRRect(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      child: Image.file(
        _selectedFile,
        height: MediaQuery.of(this.context).size.height * 0.15,
        width: MediaQuery.of(this.context).size.width * 0.3,
        fit: BoxFit.cover,
      ),
    );
  } else {
    return ClipRRect(
      borderRadius: BorderRadius.all(Radius.circular(20)),
      child: Image.asset(
        "assets/images/dd.png",
        height: MediaQuery.of(this.context).size.height * 0.15,
        width: MediaQuery.of(this.context).size.width * 0.3,
        fit: BoxFit.cover,
```

```
        ),
      );
    }
  }

  Future<String> uploadImageToFirebase(BuildContext context) async {
    String fileName = basename(_selectedFile.path);
    Reference firebaseStorageRef =
        FirebaseStorage.instance.ref().child('rescuers/$fileName');
    UploadTask uploadTask = firebaseStorageRef.putFile(_selectedFile);
    final ref = FirebaseStorage.instance.ref().child('rescuers/
$fileName');

    var image = await ref.getDownloadURL();
    return image;
  }

  void _submitForm(String image) {
    if (_formKey.currentState.validate()) {
      Rescue a = Rescue(
          id:uuid.v4(),
          name:nameController.text,
          address:addressController.text,
          amount:amountController.text,
          request:requestController.text,
          mobile:mobileNoController.text,
          image:image,);

      FormController formController = FormController();

      _showSnackbar("Submitting Feedback");

      // Submit 'feedbackForm' and save it in Google Sheets.
      formController.submitForm(a, (String response) {
        print("Response: $response");
        if (response == FormController.STATUS_SUCCESS) {
          // Feedback is saved succesfully in Google Sheets.
          _showSnackbar("Feedback Submitted");
```

```
        } else {
          // Error Occurred while saving data in Google Sheets.
          _showSnackbar("Error Occurred!");
        }
      });
    }
  }

  // Method to show snackbar with 'message'.
  _showSnackbar(String message) {
    final snackBar = SnackBar(content: Text(message));
    _scaffoldKey.currentState.showSnackBar(snackBar);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      resizeToAvoidBottomPadding: false,
      floatingActionButtonLocation:
FloatingActionButtonLocation.endFloat,
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.arrow_back),
        backgroundColor: Color(0xffBC0253),
        onPressed: () {
          BlocProvider.of<NavigationBloc>(context)
              .add(NavigationEvents.HomePageClickedEvent);
        },
      ),
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "Request a donation",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(children: [
        Image(
```

```
          image: AssetImage('assets/images/bg.png'),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          fit: BoxFit.cover,
        ),
        Positioned(
            top: MediaQuery.of(context).size.height * 0.02,
            left: MediaQuery.of(context).size.width * 0.35,
            child: Row(
              children: [
                getimageWidget(),
                Column(
                  children: [
                    IconButton(
                        icon: Icon(Icons.camera,
                            color: Color(0xffBC0253),
                            size: MediaQuery.of(context).size.height *
0.065),
                        onPressed: () {
                          getMyImage(ImageSource.camera);
                        }),
                    SizedBox(
                      height: 10,
                    ),
                    IconButton(
                        icon: Icon(Icons.photo_album,
                            color: Color(0xffBC0253),
                            size: MediaQuery.of(context).size.height *
0.065),
                        onPressed: () {
                          getMyImage(ImageSource.gallery);
                        })
                  ],
                ),
              ],
            )),
        Positioned(
          top: MediaQuery.of(context).size.height * 0.15,
```

```
            width: MediaQuery.of(context).size.width,
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: <Widget>[
                  Form(
                      key: _formKey,
                      child: Padding(
                        padding: EdgeInsets.all(16),
                        child: Column(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: <Widget>[
                            TextFormField(
                              controller: nameController,
                              validator: (value) {
                                if (value.isEmpty) {
                                  return 'Enter Valid Name';
                                }
                                return null;
                              },
                              decoration: InputDecoration(labelText:
'Name'),
                            ),

                            TextFormField(
                              controller: addressController,
                              validator: (value) {
                                if (value.isEmpty) {
                                  return 'Enter Valid address';
                                }
                                return null;
                              },
                              decoration: InputDecoration(labelText:
'address'),
                            ),
                            TextFormField(
                              controller: amountController,
                              validator: (value) {
                                if (value.isEmpty) {
```

```
                    return 'enter valid amount';
                  }
                  return null;
                },
                decoration:
                    InputDecoration(labelText: 'amount'),
              ),TextFormField(
                controller: requestController,
                validator: (value) {
                  if (value.isEmpty) {
                    return 'enter valid request';
                  }
                  return null;
                },
                decoration:
                    InputDecoration(labelText: 'request'),
              ),
              TextFormField(
                controller: mobileNoController,
                validator: (value) {
                  if (value.trim().length != 10) {
                    return 'Enter 10 Digit Mobile Number';
                  }
                  return null;
                },
                keyboardType: TextInputType.number,
                decoration: InputDecoration(
                  labelText: 'Mobile Number',
                ),
              ),
            ],
          ),
        )),
RaisedButton(
  color: Color(0xffBC0253),
  textColor: Colors.white,
  onPressed: () async {
    String image = await uploadImageToFirebase(context);
```

```
                    print(image);
                    _submitForm(image);
                  },
                  child: Text('Upload Request'),
                ),
              ]),
        )
      ]),
    );
  }
}
```

BREEDS.DART

```
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter/material.dart';
import 'package:url_launcher/url_launcher.dart';

class Breeds extends StatefulWidget with NavigationStates {
  @override
  _Breeds createState() => _Breeds();
}

class _Breeds extends State<Breeds> {
  _launchBMI() async {
  const url = 'https://tractive.com/bmi';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}



  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "Breeds",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(children: [
        Image(
          image: AssetImage('assets/images/bg.png'),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          fit: BoxFit.cover,
        ),
        Positioned(
            top: MediaQuery.of(context).size.height * 0.1,
            child: Container(
                width: MediaQuery.of(context).size.width,
                )),
                Column(children: [Row(children: [Padding(padding:
EdgeInsets.all(20),
                                child: Container(

child: ClipRRect(
                            borderRadius:
BorderRadius.all(Radius.circular(20)),
                            child: Image.network(
                              'https://images.unsplash.com/
photo-1548199973-03cce0bbc87b?
ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=800&q=60'
,
                              height: MediaQuery.of(context).size.height
* 0.2,
                              width: MediaQuery.of(context).size.width *
0.40,
                              fit: BoxFit.cover,
                            ),
                          ),
```

```
                                    ),
                ),Container()],)],)
      ]),
    );
  }
}
```

HOMEPAGE.DART

```
import 'package:Final/Pages/Adopt.dart';
import 'package:Final/Pages/Breeds.dart';
import 'package:Final/Pages/Maps.dart';
import 'package:Final/Pages/Tips.dart';
import 'package:Final/Pages/myPetChart.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:Final/lists/animalList.dart';
import 'package:flutter/material.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class HomePage extends StatefulWidget with NavigationStates {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int _selectedIndex = 0;

  List<Widget> widgetOption = [myPetChart(),Help(),Breeds(),Tips()];

  @override
  void initState() {
    // TODO: implement initState
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    onTap1(int index) {
```

```
      setState(() {
        _selectedIndex = index;
      });
    }

    return Scaffold(
      floatingActionButtonLocation:
FloatingActionButtonLocation.centerDocked,
      floatingActionButton: FloatingActionButton(
        backgroundColor: Color(0xffBC0253),
        child: Icon(Icons.add),
        shape: CircleBorder(side: BorderSide(color: Colors.white, width:
3.0)),
        onPressed: () {
          BlocProvider.of<NavigationBloc>(context)
              .add(NavigationEvents.AddPetClickedEvent);
        },
      ),
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _selectedIndex,
        iconSize: 32,
        backgroundColor: Color(0xffBC0253),
        unselectedItemColor: Colors.white,
        selectedItemColor: Colors.pink[100],
        onTap: onTap1,
        type: BottomNavigationBarType.fixed,
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'home',
            backgroundColor: Colors.white,
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.map),
            label: 'maps',
            backgroundColor: Colors.white,
          ),
          BottomNavigationBarItem(
```

```
            icon: Icon(Icons.pets),
            label: 'pets',
            backgroundColor: Color(0xffBC0253),
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.help),
            label: 'tips',
            backgroundColor: Color(0xffBC0253),
          ),
        ],
      ),
           body: widgetOption.elementAt(_selectedIndex),);
  }
}
```

MAPS.DART

```
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter/material.dart';
import 'dart:async';
import 'package:google_maps_flutter/google_maps_flutter.dart';

class Help extends StatefulWidget with NavigationStates {
  @override
  _HelpState createState() => _HelpState();
}

class _HelpState extends State<Help> {
  Completer<GoogleMapController> _controller = Completer();
  static const LatLng _center = const LatLng(45.521563, -122.677433);
  final Set<Marker> _markers = {};
  LatLng _lastMapPosition = _center;
  MapType _currentMapType = MapType.normal;

  static final CameraPosition _position1 = CameraPosition(
    bearing: 192.833,
    target: LatLng(45.521563, -122.677433),
    tilt: 59.440,
```

```
    zoom: 11.0,
  );
  Future<void> _goToPosition1() async {
    final GoogleMapController controller = await _controller.future;
    controller.animateCamera(CameraUpdate.newCameraPosition(_position1));
  }

  _onMapCreated(GoogleMapController controller) {
    _controller.complete(controller);
  }

  _onCameraMove(CameraPosition position) {
    _lastMapPosition = position.target;
  }

  _onMapTypeButtonPressed() {
    setState(() {
      _currentMapType = _currentMapType == MapType.normal
          ? MapType.satellite
          : MapType.normal;
    });
  }

  _onAddMarkerButtonPressed() {
    setState(() {
      _markers.add(Marker(
        markerId: MarkerId(_lastMapPosition.toString()),
        position: _lastMapPosition,
        infoWindow: InfoWindow(
          title: 'this is a title',
          snippet: 'This is a Snippet',
        ),
        icon: BitmapDescriptor.defaultMarker,
      ));
    });
  }

  Widget button(Function func, IconData icon) {
```

```
  return FloatingActionButton(
    onPressed: func,
    materialTapTargetSize: MaterialTapTargetSize.padded,
    backgroundColor: Color(0xffBC0253),
    child: Icon(
      icon,
      size: 36,
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Center(child: Text('Maps')),
      backgroundColor: Color(0xffBC0253),
    ),
    body: Stack(
      children: [
        GoogleMap(
          onMapCreated: _onMapCreated,
          initialCameraPosition: CameraPosition(
            target: _center,
            zoom: 11.0,
          ),
          mapType: _currentMapType,
          markers: _markers,
          onCameraMove: _onCameraMove,
        ),
        Padding(
          padding: EdgeInsets.all(16.0),
          child: Align(
            alignment: Alignment.topRight,
            child: Column(
              children: [
                button(_onMapTypeButtonPressed, Icons.map),
                SizedBox(
```

```
                    height: 16.0,
                  ),
                  button(_onAddMarkerButtonPressed, Icons.add_location),
                  SizedBox(
                    height: 16.0,
                  ),
                  button(_goToPosition1, Icons.location_searching),
                  SizedBox(
                    height: 16.0,
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

MYPETCHART.DART

```
import 'package:Final/lists/animalList.dart';
import 'package:flutter/material.dart';

class myPetChart extends StatefulWidget {
  @override
  _myPetChartState createState() => _myPetChartState();
}

class _myPetChartState extends State<myPetChart> {
  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        Image(
          image: NetworkImage(
```

```
            "https://images.pexels.com/photos/220938/pexels-
photo-220938.jpeg?auto=compress&cs=tinysrgb&dpr=2&w=500"),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          fit: BoxFit.cover,
        ),
        Positioned(
          bottom: 0,
          child: Container(
              width: MediaQuery.of(context).size.width,
              height: MediaQuery.of(context).size.height * 0.6,
              child: Stack(
                children: [
                  ClipRRect(
                    borderRadius: BorderRadius.only(
                      topLeft: Radius.circular(40),
                      topRight: Radius.circular(40),
                    ),
                    child: Image(
                      width: MediaQuery.of(context).size.width,
                      height: MediaQuery.of(context).size.height * 0.7,
                      fit: BoxFit.cover,
                      image: AssetImage('assets/images/bg.png'),
                    ),
                  )
                ],
              ),
              decoration: BoxDecoration(
                  color: Colors.amber,
                  borderRadius: BorderRadius.only(
                    topRight: Radius.circular(40.0),
                    topLeft: Radius.circular(40.0),
                  ))),
        ),
        Positioned(bottom: 0, child: AnimalList()),
      ],
    );
  }
```

```
}
```

RESCUERS.DART

```
import 'dart:ui';
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:Final/Entitites/rescuers.dart';

class Rescuers extends StatefulWidget with NavigationStates {
  @override
  _RescuersState createState() => _RescuersState();
}

class _RescuersState extends State<Rescuers> {
  List<Rescue> adoptList = List<Rescue>();

  String url =
      "https://script.google.com/macros/s/
AKfycbyrvCrRjj1a5yOiBFeDY1pMmzXHiVOzyptS7BZ8fna042dZXlA/exec";
  Future<List<Rescue>> getFeedbackList() async {
    return await http.get(url).then((response) {
      var jsonFeedback = convert.jsonDecode(response.body) as List;
      return jsonFeedback.map((json) => Rescue.fromJson(json)).toList();
    });
  }

  @override
  void initState() {
    super.initState();

    getFeedbackList().then((feedbackItems) {
      setState(() {
        this.adoptList = feedbackItems;
      });
```

```
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButtonLocation:
FloatingActionButtonLocation.endFloat,
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.arrow_back),
        backgroundColor: Color(0xffBC0253),
        onPressed: () {
          BlocProvider.of<NavigationBloc>(context)
              .add(NavigationEvents.HomePageClickedEvent);
        },
      ),
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "NGO's and Rescuers",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(
        children: [
          Image(
            image: AssetImage('assets/images/bg.png'),
            width: MediaQuery.of(context).size.width,
            height: MediaQuery.of(context).size.height,
            fit: BoxFit.cover,
          ),
          Positioned(
            top: MediaQuery.of(context).size.height * 0.1,
            child: Container(
              height: MediaQuery.of(context).size.height * 0.7,
              width: MediaQuery.of(context).size.width,
              child: ListView.builder(
                itemCount: adoptList.length,
```

```
                itemBuilder: (context, index) {
                  return Padding(
                    padding: EdgeInsets.all(10),
                    child: GestureDetector(
                      onTap: () {
                        showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return AlertDialog(
                              title: Text("Request"),
                              content: Text("$
{adoptList[index].request}"),
                              actions: [
                                FlatButton(
                                  child: Text("OK"),
                                  onPressed: ()
{Navigator.of(context).pop();},
                                ),
                              ],
                            );
                          },
                        );
                      },
                      child: Row(
                        children: [
                          ClipRRect(
                            borderRadius: BorderRadius.circular(20),
                            child: Image.network(
                              "${adoptList[index].image}",
                              width: MediaQuery.of(context).size.width *
0.4,
                              height: MediaQuery.of(context).size.height
* 0.2,
                              fit: BoxFit.cover,
                            ),
                          ),
                          Container(
```

```
                            height: MediaQuery.of(context).size.height *
0.16,
                            width: MediaQuery.of(context).size.width *
0.5,
                            decoration: BoxDecoration(
                              borderRadius: BorderRadius.only(
                                  bottomRight: Radius.circular(20),
                                  topRight: Radius.circular(20)),
                              color: Colors.white,
                            ),
                            child: Column(
                              crossAxisAlignment:
CrossAxisAlignment.start,
                              children: [
                                Padding(
                                    padding: EdgeInsets.all(5),
                                    child: Text(
                                      '${adoptList[index].name}',
                                      style: TextStyle(fontSize: 20),
                                    )),
                                Padding(
                                    padding: EdgeInsets.fromLTRB(10, 0,
5, 0),
                                    child: Text('$
{adoptList[index].address}')),
                                Padding(
                                  padding: EdgeInsets.fromLTRB(10, 5, 5,
0),
                                  child: Row(
                                    children: [
                                      Icon(Icons.phone_in_talk),
                                      Text("${adoptList[index].mobile}")
                                    ],
                                  ),
                                ),
                                Padding(
                                    padding: EdgeInsets.fromLTRB(10, 10,
5, 0),
```

```
                                    child: Text(
                                        'requested amount : $
{adoptList[index].amount}')),
                                ],
                              ),
                            )
                          ],
                        ),
                      );
                    },
                  ),
                ),
              )
            ],
          ),
        );
    }
}
```

TIPS.DART

```
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:url_launcher/url_launcher.dart';

class Tips extends StatefulWidget with NavigationStates {
  @override
  _TipsState createState() => _TipsState();
}

class _TipsState extends State<Tips> {
  _launchBMI() async {
  const url = 'https://tractive.com/bmi';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
```

```
  }
}
_launchSnacks() async {
  const url = 'https://raleighncvet.com/nutrition-weight-management/11-healthy-natural-treats-for-dogs-in-your-kitchen/';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}

_launchCvD() async {
  const url = 'https://www.purina.co.uk/cats/getting-a-new-cat/finding-the-right-cat-for-me/dog-or-cat-how-to-choose-the-right-pet-for-you';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}

_launchPotty() async {
  const url = 'https://www.humanesociety.org/resources/how-housetrain-your-dog-or-puppy';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}

_launchDeworm() async {
  const url = 'https://www.revivalanimal.com/pet-health/worming-schedule/learning-center';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
```

```
    throw 'Could not launch $url';
  }
}


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color(0xffBC0253),
        title: Text(
          "Tips and Tricks",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: Stack(children: [
        Image(
          image: AssetImage('assets/images/bg.png'),
          width: MediaQuery.of(context).size.width,
          height: MediaQuery.of(context).size.height,
          fit: BoxFit.cover,
        ),
        Positioned(
            top: MediaQuery.of(context).size.height * 0.1,
            child: Container(
                width: MediaQuery.of(context).size.width,
                child: Column(
                  children: [
                    GestureDetector(
                                          child: Card(
                        child: ListTile(title: Text('Calculate BMI'),
                          tileColor: Colors.white,
                          leading: Icon(Icons.policy),
                        ),
                      ),
                    onTap: _launchBMI,),
                    GestureDetector(onTap: _launchSnacks,
                                          child: Card(
```

```
                        child: ListTile(title: Text('Healthy all time
snacks'),
                          tileColor: Colors.white,
                          leading: Icon(Icons.policy),
                        ),
                      ),
                    ),
                    GestureDetector(onTap: _launchCvD,
                                    child: Card(
                        child: ListTile(title: Text('Cats vs Dogs'),
                          tileColor: Colors.white,
                          leading: Icon(Icons.policy),
                        ),
                      ),
                    ),GestureDetector(onTap:_launchPotty ,
                                    child: Card(
                        child: ListTile(title: Text('Potty training
101'),
                          tileColor: Colors.white,
                          leading: Icon(Icons.policy),
                        ),
                      ),
                    ),GestureDetector(onTap: _launchDeworm,
                                    child: Card(
                        child: ListTile(title: Text('Deworming and
Vaccinations'),
                          tileColor: Colors.white,
                          leading: Icon(Icons.policy),
                        ),
                      ),
                    ),
                  ],
                ))),
      ]),
    );
  }
}
```

ADOPT_PROVIDER.DART

```
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;
import 'package:Final/Entitites/Adopt_pet.dart';

/// FormController is a class which does work of saving FeedbackForm in
Google Sheets using
/// HTTP GET request on Google App Script Web URL and parses response and
sends result callback.
class FormController {

  // Google App Script Web URL.
  static const String URL =  "https://script.google.com/macros/s/
AKfycbwEl2uTHn67dlKDedEy87q_EWAt47kL4UaeAnSmQ7SmFuX0j58/exec";


  // Success Status Message
  static const STATUS_SUCCESS = "SUCCESS";

  /// Async function which saves feedback, parses [feedbackForm]
parameters
  /// and sends HTTP GET request on [URL]. On successful response,
[callback] is called.
   void submitForm(
      AdoptableAnimal feedbackForm, void Function(String) callback) async
{
    try {
      await http.post(URL, body: feedbackForm.toJson()).then((response)
async {
        if (response.statusCode == 302) {
          var url = response.headers['location'];
          await http.get(url).then((response) {
            callback(convert.jsonDecode(response.body)['status']);
          });
        } else {
          callback(convert.jsonDecode(response.body)['status']);
        }
```

```
      });
    } catch (e) {
      print(e);
    }
  }

}
```

ANIMAL_PROVIDER.DART

```
import 'package:Final/Entitites/Animal.dart';
import 'package:Final/services/firestore_service.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:uuid/uuid.dart';

class AnimalProvider with ChangeNotifier {
  final String iid;
  final firestoreService = FireStoreservice();

  String _name;
  String _type;
  String _breed;
  String _gender;
  String _id;
  String _vet1;
  String _groom;
  String _feed1;
  String _feed2;
  String _image;
  var uuid = Uuid();

  AnimalProvider(this.iid);

  String get name => _name;
  String get type => _type;
  String get breed => _breed;
  String get gender => _gender;
  String get vet1 => _vet1;
```

```
  String get groom => _groom;
  String get feed1 => _feed1;
  String get feed2 => _feed2;
  String get image => _image;

  chnageName(String val) {
    _name = val;
    notifyListeners();
  }

  chnageType(String val) {
    _type = val;
    notifyListeners();
  }

  chnageBreed(String val) {
    _breed = val;
    notifyListeners();
  }

  chnageGender(String val) {
    _gender = val;
    notifyListeners();
  }

  chnagevet(DateTime vet, DateTime groom, TimeOfDay feed1, TimeOfDay
feed2,
      String image) {
    _vet1 = vet.toString();
    _groom = groom.toString();
    _feed1 = feed1.toString();
    _feed2 = feed2.toString();
    _image = image;
    notifyListeners();
  }

  loadValues(Animal animal) {
    _name = animal.name;
```

```
    _type = animal.type;
    _breed = animal.breed;
    _gender = animal.gender;
    _id = animal.id;
    _vet1 = animal.vet1;
    _groom = animal.groom;
    _feed1 = animal.feed1;
    _feed2 = animal.feed2;
    _image = animal.image;
  }

  saveProduct() {
    print(_id);
    if (_id == null) {
      var newAnimal = Animal(
        name: name,
        type: type,
        breed: breed,
        gender: gender,
        id: uuid.v4(),
        vet1: vet1,
        groom: groom,
        feed1: feed1,
        feed2: feed2,
        image: image,
      );
      firestoreService.saveAnimal(newAnimal,iid);
    } else {
      var updateAnimal = Animal(
        name: name,
        type: _type,
        breed: breed,
        gender: gender,
        id: _id,
        vet1: vet1,
        groom: groom,
        feed1: feed1,
        feed2: feed2,
```

```
        image: image,
      );
      firestoreService.saveAnimal(updateAnimal,iid);
    }
  }

  removeAnimal(String id) {
    firestoreService.removeProduct(id,iid);
  }
}
```

RESCUE_PROVIDER.DART

```
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;
import 'package:Final/Entitites/rescuers.dart';

/// FormController is a class which does work of saving FeedbackForm in
Google Sheets using
/// HTTP GET request on Google App Script Web URL and parses response and
sends result callback.
class FormController {

  // Google App Script Web URL.
  static const String URL =  'https://script.google.com/macros/s/
AKfycbyrvCrRjj1a5yOiBFeDY1pMmzXHiVOzyptS7BZ8fna042dZXlA/exec';


  // Success Status Message
  static const STATUS_SUCCESS = "SUCCESS";

  /// Async function which saves feedback, parses [feedbackForm]
parameters
  /// and sends HTTP GET request on [URL]. On successful response,
[callback] is called.
   void submitForm(
      Rescue feedbackForm, void Function(String) callback) async {
    try {
```

```
      await http.post(URL, body: feedbackForm.toJson()).then((response)
async {
        if (response.statusCode == 302) {
          var url = response.headers['location'];
          await http.get(url).then((response) {
            callback(convert.jsonDecode(response.body)['status']);
          });
        } else {
          callback(convert.jsonDecode(response.body)['status']);
        }
      });
    } catch (e) {
      print(e);
    }
  }

}
```

FIRESTORE_SERVICE.DART

```
import 'dart:io';

import 'package:Final/Entitites/Animal.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:device_info/device_info.dart';

class FireStoreservice {


Future<String> _getId() async {

    var deviceInfo = DeviceInfoPlugin();
    if (Platform.isIOS) {
      // import 'dart:io'
      var iosDeviceInfo = await deviceInfo.iosInfo;
      return iosDeviceInfo.identifierForVendor; // unique ID on iOS
    } else {
      var androidDeviceInfo = await deviceInfo.androidInfo;
```

```
      return androidDeviceInfo.androidId; // unique ID on Android
    }
  }
  FirebaseFirestore _db = FirebaseFirestore.instance;
  Future<void> saveAnimal(Animal animal,String iid) async {
    _db.collection(iid).doc(animal.id).set(animal.toMap());
  }

  Stream<List<Animal>> getAnimals(String iid) {
    return _db.collection(iid).snapshots().map((snapshot) =>
        snapshot.docs
            .map((document) => Animal.fromFireStore(document.data()))
            .toList());
  }

  Future<void> removeProduct(String id,String iid) {
    return _db.collection(iid).doc(id).delete();
  }
}
```

MENU_ITEM.DART

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

class MenuItem extends StatelessWidget {
  final String title;
  final Color color;
  final Function func;
  final double width;
  final double height;

  const MenuItem(
      {Key key, this.title, this.color, this.func, this.width,
this.height})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
```

```
    return GestureDetector(
      onTap: func,
      child: Container(
        width: width,
        height: height /9 ,
        padding: EdgeInsets.only(left: 30),
        alignment: Alignment.centerLeft,
        child: FittedBox(
          fit: BoxFit.fitWidth,
          child: Text(
            title,
            style: TextStyle(color: color, fontSize: 45, letterSpacing:
2),
          ),
        ),
      ),
    );
  }
}
```

SIDEBAR_LAYOUT.DART

```
import 'package:Final/Pages/homepage.dart';
import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:Final/sidebar/sidebar.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class SideBarLayout extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: BlocProvider<NavigationBloc>(
        create: (context) => NavigationBloc(),
        child: Stack(
          children: <Widget>[
            BlocBuilder<NavigationBloc, NavigationStates>(
                builder: (context, navigationState) {
```

```
              return navigationState as Widget;
            }),
            SideBar(),
          ],
        ),
      ),
    );
  }
}
```

SIDEBAR.DART

```
import 'dart:async';

import 'package:Final/bloc.navigation_bloc/navigation_bloc.dart';
import 'package:Final/sidebar/menu_item.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:rxdart/rxdart.dart';
import 'menu_item.dart';

class SideBar extends StatefulWidget {
  @override
  _SideBarState createState() => _SideBarState();
}

class _SideBarState extends State<SideBar>
    with SingleTickerProviderStateMixin<SideBar> {
  AnimationController _animationController;
  StreamController<bool> isSideBarOpenedSreamcontroller;
  Stream<bool> isSideBaropenedStream;
  StreamSink<bool> isSideBarOpenedSink;
  final _animationDuration = const Duration(milliseconds: 300);

  @override
  void initState() {
    super.initState();
    _animationController =
```

```
        AnimationController(vsync: this, duration: _animationDuration);
    isSideBarOpenedSreamcontroller = PublishSubject<bool>();
    isSideBaropenedStream = isSideBarOpenedSreamcontroller.stream;
    isSideBarOpenedSink = isSideBarOpenedSreamcontroller.sink;
  }

  void dispose() {
    _animationController.dispose();
    isSideBarOpenedSreamcontroller.close();
    isSideBarOpenedSink.close();
    super.dispose();
  }

  void onIconPressed() {
    final animationStatus = _animationController.status;
    final isAnimationCompleted = animationStatus ==
AnimationStatus.completed;

    if (isAnimationCompleted) {
      isSideBarOpenedSink.add(false);
      _animationController.reverse();
    } else {
      isSideBarOpenedSink.add(true);

      _animationController.forward();
    }
  }

  Widget build(BuildContext context) {
    final width = MediaQuery.of(context).size.width;
    final height = MediaQuery.of(context).size.height;
    print(width);
    print(width - 0.8 * width);
    return StreamBuilder<bool>(
      initialData: false,
      stream: isSideBaropenedStream,
      builder: (context, isSideBarOpenedAsync) {
        return AnimatedPositioned(
```

```
          duration: _animationDuration,
          top: 0,
          bottom: 0,
          left: isSideBarOpenedAsync.data ? 0 : -width,
          right: isSideBarOpenedAsync.data
              ? width - 0.8 * width
              : width - width / 6.9,
          child: Row(
            children: [
              Expanded(
                child: Container(
                  child: Column(
                    children: [
                      SafeArea(
                        child: SizedBox(
                          height: height / 4.48,
                        ),
                      ),
                      Divider(
                        height: 64,
                        thickness: 0.5,
                        color: Colors.white,
                        indent: 32,
                        endIndent: 32,
                      ),
                      MenuItem(
                        height:MediaQuery.of(context).size.height-
height / 4.48-kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'adopt',
                        color: Color(0xFFBA2D65),
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.AdoptClickedEvent);
                        },
                      ),
```

```
MenuItem( height:MediaQuery.of(context).size.height-height / 4.48-
kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'donate',
                        color: Colors.white,
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.RescuersClickedEvent)
;
                        },
                      ),

MenuItem( height:MediaQuery.of(context).size.height-height / 4.48-
kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'breeds',
                        color: Color(0xFFBA2D65),
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.BreedsClickedEvent);
                        },
                      ),


MenuItem( height:MediaQuery.of(context).size.height-height / 4.48-
kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'request',
                        color: Colors.white,
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.DonateClickedEvent);
                        },
                      ),
```

```
MenuItem( height:MediaQuery.of(context).size.height-height / 4.48-
kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'maps',
                        color: Color(0xFFBA2D65),
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.HelpClickedEvent);
                        },
                      ),

MenuItem( height:MediaQuery.of(context).size.height-height / 4.48-
kToolbarHeight,
                        width:MediaQuery.of(context).size.width*0.8,
                        title: 'put up',
                        color: Colors.white,
                        func: () {
                          onIconPressed();
                          BlocProvider.of<NavigationBloc>(context)
                              .add(NavigationEvents.AboutUsClickedEvent);
                        },
                      ),
                    ],
                  ),
                  color: Color(0xFFF48FB1),
                ),
              ),
              Align(
                alignment: Alignment(0, -0.8),
                child: GestureDetector(
                  onTap: () {
                    onIconPressed();
                  },
                  child: ClipPath(
                    clipper: CustomMenuClipper(),
                    child: Container(
```

```
                        width: width / 6.9,
                        height: height / 6.4,
                        color: Color(0xFFF48FB1),
                        alignment: Alignment.centerLeft,
                        child: AnimatedIcon(
                            icon: AnimatedIcons.menu_close,
                            progress: _animationController.view,
                            color: Colors.white,
                            size: width / 6.9 - 15),
                      ),
                    ),
                  ),
                )
              ],
            ),
          );
        },
      );
    }
}

class CustomMenuClipper extends CustomClipper<Path> {
  @override
  Path getClip(Size size) {
    Paint paint = Paint();
    paint.color = Colors.white;
    Path path = Path();
    final width = size.width;
    final height = size.height;
    path.moveTo(0, 0);
    path.quadraticBezierTo(0, 8, 10, 16);
    path.quadraticBezierTo(width - 1, height / 2 - 20, width, height /
2);
    path.quadraticBezierTo(width + 1, height / 2 + 20, 10, height - 16);
    path.quadraticBezierTo(0, height - 8, 0, height);
    path.close();

    return path;
```

```
  }

  @override
  bool shouldReclip(CustomClipper<Path> oldClipper) {
    return true;
  }
}
```

MAIN.DART

```
import 'dart:io';

import 'package:Final/services/firestore_service.dart';
import 'package:Final/sidebar/sidebar_layout.dart';
import 'package:device_info/device_info.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'providers/animal_provider.dart';

void main() async {
  String deviceId;
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  Future<String> _getId() async {
    var deviceInfo = DeviceInfoPlugin();
    if (Platform.isIOS) {
      // import 'dart:io'
      var iosDeviceInfo = await deviceInfo.iosInfo;
      return iosDeviceInfo.identifierForVendor; // unique ID on iOS
    } else {
      var androidDeviceInfo = await deviceInfo.androidInfo;
      return androidDeviceInfo.androidId; // unique ID on Android
    }
  }

  deviceId = await _getId();

  runApp(MyApp(deviceId:deviceId));
```

```
}

class MyApp extends StatelessWidget {
  final String deviceId;

  const MyApp({Key key, this.deviceId}) : super(key: key);



  @override
  Widget build(BuildContext context) {
    final firestoreService = FireStoreservice();

    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (context) => AnimalProvider(deviceId),
        ),
        StreamProvider(create: (context) =>
firestoreService.getAnimals(deviceId)),
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        home: SideBarLayout(),
      ),
    );
  }
}
```

WIDGET_TEST.DART

```
// This is a basic Flutter widget test.
//
// To perform an interaction with a widget in your test, use the
WidgetTester
// utility that Flutter provides. For example, you can send tap and
scroll
```

```
// gestures. You can also use WidgetTester to find child widgets in the
widget
// tree, read text, and verify that the values of widget properties are
correct.

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

import 'package:Final/main.dart';

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester)
async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

PUBSPEC.YAML

```
name: Final
description: A new Flutter project.

# The following line prevents the package from being accidentally
published to
# pub.dev using `pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev
```

```
# The following defines the version and build number for your
application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used
as versionCode.
# Read more about Android versioning at https://developer.android.com/
studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-
number used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/
Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
version: 1.0.0+1

environment:
  sdk: ">=2.0.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.3
  flutter_bloc: ^2.1.1
  google_maps_flutter: ^0.5.33
  path: ^1.7.0
  http: ^0.12.0+3
  firebase_core: ^0.5.0+1
  cloud_firestore: ^0.14.1+3
  provider: ^3.2.0
  uuid: ^2.2.2
  image_picker: ^0.6.7+12
```

```
  image_cropper: ^1.3.1
  intl: ^0.16.1
  flutter_local_notifications: ^3.0.0+1
  rxdart: ^0.22.0
  path_provider: ^1.6.22
  ffi: ^0.1.3
  device_info: ^0.4.0+4
  permission_handler: ^5.0.1+1
  firebase_storage: ^5.0.0
  url_launcher: ^5.7.10




dev_dependencies:
  flutter_test:
    sdk: flutter

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/images/bg.png
    - assets/images/dd.png

  # An image asset can refer to one or more resolution-specific
"variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware.
```

```
# For details regarding adding assets from package dependencies, see
# https://flutter.dev/assets-and-images/#from-packages

# To add custom fonts to your application, add a fonts section here,
# in this "flutter" section. Each entry in this list should have a
# "family" key with the font family name, and a "fonts" key with a
# list giving the asset and other descriptors for the font. For
# example:
fonts:
   - family: blue
     fonts:
       - asset: assets/Blueberry .ttf
#       - asset: fonts/Schyler-Italic.ttf
#         style: italic
#   - family: Trajan Pro
#     fonts:
#       - asset: fonts/TrajanPro.ttf
#       - asset: fonts/TrajanPro_Bold.ttf
#         weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

CONCLUSION

Henceforth, pet's point will act as an apt solution to individual pet owners, as well as help in providing shelter to abandoned and local animals. Also provide assistance in increasing the reach of rescuers and NGO's to common public

FUTURE SCOPE

- Add shop for pet related purchases
- Add a page for owners to share their pet photos and status
- Improve the usability of google maps api
- Refinement in UI

REFERENCES

- www.udemy.com
- www.youtube.com
- www.google.co.in
- www.filledstacks.com
- www.github.org
- www.pub.dev
- www.stackoverflow.com