import 'dart:async';  
import 'dart:io';  
import 'package:http/http.dart' as http;  
import 'package:bubble/bubble.dart';  
import 'package:cached\_network\_image/cached\_network\_image.dart';  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:firebase\_storage/firebase\_storage.dart';  
import 'package:flutter/material.dart';  
import 'package:chat\_app/const.dart';  
import 'package:image\_picker\_saver/image\_picker\_saver.dart' as Img\_picker;  
import 'package:social\_share\_plugin/social\_share\_plugin.dart';  
import 'package:chat\_app/widget/loading.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
import 'package:image\_picker/image\_picker.dart';  
import 'package:intl/intl.dart';  
import 'package:shared\_preferences/shared\_preferences.dart';  
import 'package:back\_button\_interceptor/back\_button\_interceptor.dart';  
  
  
  
import 'first\_screen.dart';  
  
class Chat extends StatelessWidget {  
 final String peerId;  
 final String peerAvatar;  
 final String peerName;  
 final String peerAboutMe;  
  
 Chat({Key key, @required this.peerId, @required this.peerAvatar, @required this.peerName,@required this.peerAboutMe})  
 : super(key: key);  
  
 @override  
 Widget build(BuildContext context) {  
 return ChatScreen(  
 peerId: peerId,  
 peerAvatar: peerAvatar,  
 peerName: peerName,  
 peerAboutMe:peerAboutMe,  
 );  
 }  
}  
  
class ChatScreen extends StatefulWidget {  
 final String peerId;  
 final String peerAvatar;  
 final String peerName;  
 final String peerAboutMe;  
 ChatScreen({Key key, @required this.peerId, @required this.peerAvatar, @required this.peerName, @required this.peerAboutMe})  
 : super(key: key);  
  
 @override  
 State createState() =>  
 ChatScreenState(peerId: peerId, peerAvatar: peerAvatar,peerName: peerName, peerAboutMe:peerAboutMe);  
}  
  
class ChatScreenState extends State<ChatScreen> {  
 ChatScreenState({Key key, @required this.peerId, @required this.peerAvatar,@required this.peerName, @required this.peerAboutMe});  
  
 String peerId;  
 String peerAvatar;  
 String peerName;  
 String peerAboutMe;  
 String id;  
 int flag=0;  
 List<QueryDocumentSnapshot> listMessage = new List.from([]);  
 int \_limit = 20;  
 final int \_limitIncrement = 20;  
 String groupChatId;  
 SharedPreferences prefs;  
  
 File imageFile;  
 bool isLoading;  
 String imageUrl;  
  
 final TextEditingController textEditingController = TextEditingController();  
 final ScrollController listScrollController = ScrollController();  
 final FocusNode focusNode = FocusNode();  
  
 \_scrollListener() {  
 if (listScrollController.offset >=  
 listScrollController.position.maxScrollExtent &&  
 !listScrollController.position.outOfRange) {  
 print("reach the bottom");  
 setState(() {  
 print("reach the bottom");  
 \_limit += \_limitIncrement;  
 });  
 }  
 if (listScrollController.offset <=  
 listScrollController.position.minScrollExtent &&  
 !listScrollController.position.outOfRange) {  
 print("reach the top");  
 setState(() {  
 print("reach the top");  
 });  
 }  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 BackButtonInterceptor.*add*(myInterceptor);  
 listScrollController.addListener(\_scrollListener);  
 groupChatId = '';  
 isLoading = false;  
 imageUrl = '';  
 readLocal();  
 }  
  
 @override  
 void dispose() {  
 BackButtonInterceptor.*remove*(myInterceptor);  
 super.dispose();  
 }  
  
 bool myInterceptor(bool stopDefaultButtonEvent, RouteInfo info) {  
 print("BACK BUTTON!"); // Do some stuff.  
 FirebaseFirestore.*instance* .collection('users')  
 .doc(id)  
 .update({'chattingWith': null});  
 Navigator.*of*(context).push(  
 MaterialPageRoute(  
 builder: (context) {  
 return FirstScreen(userId: id);  
 },  
 ),  
 );// Do some stuff.  
 return true;  
 }  
  
 int calculateDifference(DateTime date) {  
 DateTime now = DateTime.now();  
 return DateTime(date.year, date.month, date.day).difference(DateTime(now.year, now.month, now.day)).inDays;  
 }  
  
 readLocal() async {  
 prefs = await SharedPreferences.*getInstance*();  
 id = prefs.getString('id') ?? '';  
 if (id.hashCode <= peerId.hashCode) {  
 groupChatId = '$id-$peerId';  
 } else {  
 groupChatId = '$peerId-$id';  
 }  
  
 FirebaseFirestore.*instance* .collection('users')  
 .doc(id)  
 .update({'chattingWith': peerId});  
 setState(() {});  
 }  
  
 Future getImage() async {  
 ImagePicker imagePicker = ImagePicker();  
 PickedFile pickedFile;  
  
 pickedFile = await imagePicker.getImage(source: ImageSource.gallery);  
 imageFile = File(pickedFile.path);  
  
 if (imageFile != null) {  
 setState(() {  
 isLoading = true;  
 });  
 uploadFile();  
 }  
 }  
  
 Future uploadFile() async {  
 String fileName = DateTime.now().millisecondsSinceEpoch.toString();  
 StorageReference reference = FirebaseStorage.*instance*.ref().child(fileName);  
 StorageUploadTask uploadTask = reference.putFile(imageFile);  
 StorageTaskSnapshot storageTaskSnapshot = await uploadTask.onComplete;  
 storageTaskSnapshot.ref.getDownloadURL().then((downloadUrl) {  
 imageUrl = downloadUrl;  
 setState(() {  
 isLoading = false;  
 onSendMessage(imageUrl, 1);  
 });  
 }, onError: (err) {  
 setState(() {  
 isLoading = false;  
 });  
 Fluttertoast.*showToast*(msg: 'This file is not an image');  
 });  
 }  
  
 void onSendMessage(String content, int type) {  
 // type: 0 = text, 1 = image, 2 = sticker  
 if (content.trim() != '') {  
 textEditingController.clear();  
  
 var documentReference = FirebaseFirestore.*instance* .collection('messages')  
 .doc(groupChatId)  
 .collection(groupChatId)  
 .doc(DateTime.now().millisecondsSinceEpoch.toString());  
  
 FirebaseFirestore.*instance*.runTransaction((transaction) async {  
 transaction.set(  
 documentReference,  
 {  
 'idFrom': id,  
 'idTo': peerId,  
 'timestamp': DateTime.now().millisecondsSinceEpoch.toString(),  
 'content': content,  
 'type': type  
 },  
 );  
 });  
 listScrollController.animateTo(0.0,  
 duration: Duration(milliseconds: 300), curve: Curves.*easeOut*);  
 } else {  
 Fluttertoast.*showToast*(  
 msg: 'Nothing to send',  
 backgroundColor: Colors.*white54*,  
 textColor: Colors.*black*);  
 }  
 }  
  
 Widget buildDay(int index, DocumentSnapshot document, DocumentSnapshot document1)  
 {  
 String cd = DateFormat('dd MMMM yyyy').format(DateTime.fromMillisecondsSinceEpoch(int.*parse*(document.data()['timestamp'])));  
 String whichDay = DateFormat('dd MMMM yyyy').format(DateTime.fromMillisecondsSinceEpoch(int.*parse*(document1.data()['timestamp'])));  
 if(whichDay == cd){  
 return Container(  
 child:buildItem(index, document),  
 );  
 }  
 else {  
 return Column(  
 children: <Widget>[  
 Container(  
 padding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),  
 margin: EdgeInsets.only(  
 top: 10.0, bottom: 5.0, left: 0.0, right: 0.0),  
 decoration: BoxDecoration(  
 color: greyColor2,  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 ),  
 child: Text(cd)  
 ),  
 Container(  
 child: buildItem(index, document),  
 )  
 ],  
 );  
 }  
 }  
 Widget buildDay1(int index, DocumentSnapshot document)  
 {  
 String cd = DateFormat('dd MMMM yyyy').format(DateTime.fromMillisecondsSinceEpoch(int.*parse*(document.data()['timestamp'])));  
 return Column(  
 children: <Widget>[  
 Container(  
 padding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),  
 margin: EdgeInsets.only(  
 top: 10.0, bottom: 5.0, left: 0.0, right: 0.0),  
 decoration: BoxDecoration(  
 color: greyColor2,  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 ),  
 child: Text(cd)  
 ),  
 Container(  
 child: buildItem(index, document),  
 )  
 ],  
 );  
 }  
  
 Widget buildItem(int index, DocumentSnapshot document) {  
  
 double pixelRatio = MediaQuery.*of*(context).devicePixelRatio;  
 double px = 1 / pixelRatio;  
  
 BubbleStyle styleSomebody = BubbleStyle(  
 //nip: BubbleNip.leftTop,  
 color: greyColor2,  
 //elevation: 1 \* px,  
 //margin: BubbleEdges.only(top: 3.0, right: 50.0),  
 //alignment: Alignment.topLeft,  
 );  
 BubbleStyle styleMe = BubbleStyle(  
 //nip: BubbleNip.rightTop,  
 color: greyColor,  
 // elevation: 1 \* px,  
 //margin: BubbleEdges.only(top: 3.0, left: 50.0),  
 //alignment: Alignment.topRight,  
 );  
  
 if (document.data()['idFrom'] == id) {  
 // Right (my message)  
 return Row(  
 children: <Widget>[  
 document.data()['type'] == 0  
 // Text  
 ? Flexible(  
 child: FlatButton(  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.end,  
 mainAxisAlignment: MainAxisAlignment.end,  
 children: <Widget>[  
 Container(  
 child: Text(  
 DateFormat('hh:mm a').format(  
 DateTime.fromMillisecondsSinceEpoch(  
 int.*parse*(document.data()['timestamp']))),  
 style: TextStyle(  
 fontSize: 13, color: Colors.*black*.withOpacity(0.8)),  
 ),  
 ),  
 Flexible(  
 child:Bubble(  
 style: styleMe,  
 child:Text(  
 document.data()['content'],  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*normal*),  
 ),  
 ))  
 ]),  
 onLongPress: () {deleteBuild(context,document.id);}  
 ),  
 )  
 : Container(  
 child: FlatButton(  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.end,  
 mainAxisAlignment: MainAxisAlignment.end,  
 children: <Widget>[  
 Container(  
 child: Text(  
 DateFormat('hh:mm a').format(  
 DateTime.fromMillisecondsSinceEpoch(  
 int.*parse*(document.data()['timestamp']))),  
 style: TextStyle(  
 fontSize: 13, color: Colors.*black*.withOpacity(0.8)),  
 ),  
 ),  
 Container(  
 child: FlatButton(  
 child: Material(  
 child: CachedNetworkImage(  
 placeholder: (context, url) => Container(  
 child: CircularProgressIndicator(  
 valueColor: AlwaysStoppedAnimation<Color>(  
 themeColor),  
 ),  
 width: 200.0,  
 height: 200.0,  
 padding: EdgeInsets.all(70.0),  
 decoration: BoxDecoration(  
 color: greyColor2,  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 ),  
 ),  
 errorWidget: (context, url, error) =>  
 Material(  
 child: Image.asset(  
 'images/img\_not\_available.jpeg',  
 width: 200.0,  
 height: 200.0,  
 fit: BoxFit.cover,  
 ),  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 clipBehavior: Clip.hardEdge,  
 ),  
 imageUrl: document.data()['content'],  
 width: 200.0,  
 height: 200.0,  
 fit: BoxFit.cover,  
 ),  
 borderRadius:  
 BorderRadius.all(Radius.circular(8.0)),  
 clipBehavior: Clip.hardEdge,  
 ),  
 onPressed: () {  
  
 imageBuild(context,document.data()['content']);  
 },  
 padding: EdgeInsets.all(0),  
 ),  
 margin: EdgeInsets.only(left: 10.0),  
 )],  
 ),  
 onLongPress: () {deleteBuild(context,document.id);}  
 )),  
 ],  
 mainAxisAlignment: MainAxisAlignment.end,  
 );  
 } else {  
 // Left (peer message)  
 return Container(  
 child: Column(  
 children: <Widget>[  
 Row(  
 children: <Widget>[  
 document.data()['type'] == 0  
 ? Flexible(  
 child: FlatButton(  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.end,  
 mainAxisAlignment: MainAxisAlignment.start,  
 children: <Widget>[  
 Flexible(  
 child:Bubble(  
 style: styleSomebody,  
 child:Text(  
 document.data()['content'],  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*normal*),  
 ),  
 )),  
 Container(  
 child: Text(  
 DateFormat('hh:mm a').format(  
 DateTime.fromMillisecondsSinceEpoch(  
 int.*parse*(document.data()['timestamp']))),  
 style: TextStyle(  
 fontSize: 13, color: Colors.*black*.withOpacity(0.8)),  
 ),  
 ),  
  
 ]),  
 onLongPress: () {deleteBuild(context,document.id);}  
 )  
 )  
 : Container(  
 child: FlatButton(  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.end,  
 mainAxisAlignment: MainAxisAlignment.start,  
 children: <Widget>[  
 Container(  
 child: FlatButton(  
 child: Material(  
 child: CachedNetworkImage(  
 placeholder: (context, url) => Container(  
 child: CircularProgressIndicator(  
 valueColor: AlwaysStoppedAnimation<Color>(  
 themeColor),  
 ),  
 width: 200.0,  
 height: 200.0,  
 padding: EdgeInsets.all(70.0),  
 decoration: BoxDecoration(  
 color: greyColor2,  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 ),  
 ),  
 errorWidget: (context, url, error) =>  
 Material(  
 child: Image.asset(  
 'images/img\_not\_available.jpeg',  
 width: 200.0,  
 height: 200.0,  
 fit: BoxFit.cover,  
 ),  
 borderRadius: BorderRadius.all(  
 Radius.circular(8.0),  
 ),  
 clipBehavior: Clip.hardEdge,  
 ),  
 imageUrl: document.data()['content'],  
 width: 200.0,  
 height: 200.0,  
 fit: BoxFit.cover,  
 ),  
 borderRadius:  
 BorderRadius.all(Radius.circular(8.0)),  
 clipBehavior: Clip.hardEdge,  
 ),  
 onPressed: () {  
 imageBuild(context,document.data()['content']);  
 },  
 padding: EdgeInsets.all(0),  
 ),  
 margin: EdgeInsets.only(left: 10.0),  
 ),  
 Container(  
 child: Text(  
 DateFormat('hh:mm a').format(  
 DateTime.fromMillisecondsSinceEpoch(  
 int.*parse*(document.data()['timestamp']))),  
 style: TextStyle(  
 fontSize: 13, color: Colors.*black*.withOpacity(0.8)),  
 ),  
 ),],  
 ),  
 onLongPress: () {deleteBuild(context,document.id);}  
 )),  
 ],  
 ),  
 ],  
 crossAxisAlignment: CrossAxisAlignment.start,  
 ),  
 margin: EdgeInsets.only(bottom: 10.0),  
 );  
 }  
 }  
  
 bool isLastMessageLeft(int index) {  
 if ((index > 0 &&  
 listMessage != null &&  
 listMessage[index - 1].data()['idFrom'] == id) ||  
 index == 0) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 bool isLastMessageRight(int index) {  
 if ((index > 0 &&  
 listMessage != null &&  
 listMessage[index - 1].data()['idFrom'] != id) ||  
 index == 0) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 Future<bool> onBackPress() {  
 FirebaseFirestore.*instance* .collection('users')  
 .doc(id)  
 .update({'chattingWith': null});  
 Navigator.*of*(context).push(  
 MaterialPageRoute(  
 builder: (context) {  
 return FirstScreen(userId: id);  
 },  
 ),  
 );// Do some stuff.  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 centerTitle: true,  
 title: Text(  
 peerName,  
 style: new TextStyle(color: Colors.*black*,  
 fontSize: 20)),  
 actions: <Widget>[  
 PopupMenuButton<String>(  
 onSelected: handleClick,  
 itemBuilder: (BuildContext context) {  
 return {'Delete All'}.map((String choice) {  
 return PopupMenuItem<String>(  
 value: choice,  
 child: Text(choice),  
 );  
 }).toList();  
 },  
 ),  
 ],  
 ),  
 body: WillPopScope(  
 child: Stack(  
 children: <Widget>[  
 Column(  
 children: <Widget>[  
 // List of messages  
 buildListMessage(),  
 // Input content  
 buildInput(),  
 ],  
 ),  
 // Loading  
 buildLoading()  
 ],  
 ),  
 onWillPop: onBackPress,  
 ));  
 }  
  
 Widget buildLoading() {  
 return Positioned(  
 child: isLoading ? const Loading() : Container(),  
 );  
 }  
  
 Widget buildInput() {  
 return Container(  
 child: Row(  
 children: <Widget>[  
 // Button send image  
 Material(  
 child: Container(  
 margin: EdgeInsets.symmetric(horizontal: 1.0),  
 child: IconButton(  
 icon: Icon(Icons.*photo\_camera*),  
 onPressed: getImage,  
 color: Colors.*black87*,  
 ),  
 ),  
 color: greyColor2,  
 ),  
 // Edit text  
 Flexible(  
 child: Container(  
 child: TextField(  
 onSubmitted: (value) {  
 onSendMessage(textEditingController.text, 0);  
 },  
 style: TextStyle(color: primaryColor, fontSize: 15.0),  
 controller: textEditingController,  
 decoration: InputDecoration.collapsed(  
 hintText: 'メッセージを入力',  
 hintStyle: TextStyle(color: Colors.*black54*),  
 ),  
 focusNode: focusNode,  
 ),  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.only(  
 bottomLeft: Radius.circular(22.0),  
 bottomRight: Radius.circular(22.0),  
 topLeft: Radius.circular(22.0),  
 topRight: Radius.circular(22.0)))),  
 ),  
 // Button send message  
 Material(  
 child: Container(  
 margin: EdgeInsets.symmetric(horizontal: 8.0),  
 child: IconButton(  
 icon: Icon(Icons.*play\_arrow*),  
 onPressed: () => onSendMessage(textEditingController.text, 0),  
 color: Colors.*black87*,  
 ),  
 ),  
 color: greyColor2,  
 ),  
 ],  
 ),  
 width: double.*infinity*,  
 height: 50.0,  
 decoration: BoxDecoration(  
 border: Border(top: BorderSide(color: greyColor2, width: 0.5)),  
 color: Colors.*white*),  
 );  
 }  
  
 Widget buildListMessage() {  
 return Flexible(  
 child: groupChatId == ''  
 ? Center(  
 child: CircularProgressIndicator(  
 valueColor: AlwaysStoppedAnimation<Color>(themeColor)))  
 : StreamBuilder(  
 stream: FirebaseFirestore.*instance* .collection('messages')  
 .doc(groupChatId)  
 .collection(groupChatId)  
 .orderBy('timestamp', descending: true)  
 .limit(\_limit)  
 .snapshots(),  
 builder: (context, snapshot) {  
 if (!snapshot.hasData) {  
 return Center(  
 child: CircularProgressIndicator(  
 valueColor:  
 AlwaysStoppedAnimation<Color>(themeColor)));  
 } else {  
 listMessage.addAll(snapshot.data.documents);  
 int itemLength = snapshot.data.documents.length - 1;  
 return ListView.builder(  
 padding: EdgeInsets.all(10.0),  
 itemCount: snapshot.data.documents.length,  
 itemBuilder: (context, index) {  
 if(index != itemLength){  
 return buildDay(index, snapshot.data.documents[index], snapshot.data.documents[index+1]);}  
 else{  
 return buildDay1(index, snapshot.data.documents[index]);  
 }  
 },  
 reverse: true,  
 controller: listScrollController,  
 );  
 }  
 },  
 ),  
 );  
 }  
  
 void handleClick(String value){  
 switch (value) {  
 case 'Delete All':  
 {  
 deleteAllBuild(context);  
 }  
 break;  
 }  
 }  
  
 void deleteAll() async {  
  
 var collectionReference = await FirebaseFirestore.*instance*.collection("messages");  
 Future<QuerySnapshot> group = collectionReference.doc(groupChatId).collection(groupChatId).get();  
 group.then((value) {  
 value.docs.forEach((element) {  
 collectionReference  
 .doc(groupChatId)  
 .collection(groupChatId)  
 .doc(element.id)  
 .delete()  
 .then((value) => print("success"));  
 });  
 });  
 }  
  
 deleteAllBuild(BuildContext context) {  
  
 Widget okButton = FlatButton(  
 child: Text("OK"),  
 onPressed: () {  
 deleteAll();  
 Navigator.*of*(context, rootNavigator: true).pop();  
 },  
 );  
  
 Widget cancelButton = FlatButton(  
 child: Text("Cancel"),  
 onPressed: () {  
 Navigator.*of*(context, rootNavigator: true).pop();  
 },  
 );  
  
 AlertDialog alert = AlertDialog(  
 content: Text("Do you want to delete all the messages?"),  
 actions: [  
 cancelButton,  
 okButton,  
 ],  
 );  
  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return alert;  
 },  
 );  
 }  
  
 void onDelete(String timestamp) async{  
  
 var documentReference = FirebaseFirestore.*instance*;  
 documentReference.collection("messages").doc(groupChatId).collection(groupChatId)  
 .doc(timestamp).delete();  
 }  
  
 deleteBuild(BuildContext context,String timestamp){  
 Widget okButton = FlatButton(  
 child: Text("OK"),  
 onPressed: () {  
 onDelete(timestamp);  
 Navigator.*of*(context, rootNavigator: true).pop();  
 },  
 );  
  
 Widget cancelButton = FlatButton(  
 child: Text("Cancel"),  
 onPressed: () {  
 Navigator.*of*(context, rootNavigator: true).pop();  
 },  
 );  
  
 AlertDialog alert = AlertDialog(  
 content: Text("Do you want to delete this message?"),  
 actions: [  
 cancelButton,  
 okButton,  
 ],  
 );  
  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return alert;  
 },  
 );  
 }  
  
 imageBuild(BuildContext context,String url){  
  
 Widget cancelButton = FlatButton(  
 child: Text("キャンセル",  
 style:TextStyle(color: Colors.*grey*)),  
 onPressed: () {  
 Navigator.*of*(context, rootNavigator: true).pop();  
 },  
 );  
  
 AlertDialog alert = AlertDialog(  
 content:Container(  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 Flexible(  
 child:Text(" シェアする写真は\nこちらでよろしいでしょうか")  
 ),  
 SizedBox(height: 10),  
 Container(  
 constraints: new BoxConstraints.expand(  
 width: 600.0,  
 height: 250.0,  
 ),  
 decoration: new BoxDecoration(  
 image: new DecorationImage(  
 image: NetworkImage("$url"),  
 fit: BoxFit.fill,  
 ),  
 ),  
 child: new Stack(  
 children: <Widget>[  
 new Positioned(  
 right: 0.0,  
 bottom: 0.0,  
 child: Image.asset('assets/AppLogo.png',scale: 15.0),  
 ),  
 ],  
 )  
 ),  
 SizedBox(height: 10),  
 FlatButton(  
 child: Row(  
 children: [  
 SizedBox(width: 5),  
 Image(  
 image: AssetImage("assets/instagram.png"),  
 height: 22.0,  
 ),  
 SizedBox(width: 10),  
 Text('Instagramでシェア',  
 style:TextStyle(  
 fontSize: 20,  
 color: Colors.*white*,  
 ),),  
 ],  
 ),  
 onPressed: () async {  
 var response = await http.get(url);  
 debugPrint(response.statusCode.toString());  
 var filePath = await Img\_picker.ImagePickerSaver.*saveFile*(  
 fileData: response.bodyBytes);  
 var savedFile= File.fromUri(Uri.file(filePath));  
 File file = savedFile;  
 await SocialSharePlugin.*shareToFeedInstagram*(path: file.path);  
 },  
 color: Colors.*grey*,  
 ),  
 FlatButton(  
 child: Row(  
 children: [  
 SizedBox(width: 5),  
 Image(  
 image: AssetImage("assets/facebook.png"),  
 height: 22.0,  
 ),  
 SizedBox(width: 10),  
 Text('Facebookでシェア',  
 style:TextStyle(  
 fontSize: 20,  
 color: Colors.*white*,  
 ),),  
 ],  
 ),  
 onPressed: () async {  
 var response = await http.get(url);  
 debugPrint(response.statusCode.toString());  
 var filePath = await Img\_picker.ImagePickerSaver.*saveFile*(  
 fileData: response.bodyBytes);  
 var savedFile= File.fromUri(Uri.file(filePath));  
 File file = savedFile;  
 await SocialSharePlugin.*shareToFeedFacebook*(path: file.path);  
 },  
 color: Colors.*grey*,  
 ),  
 FlatButton(  
 child: Row(  
 children: [  
 SizedBox(width: 5),  
 Image(  
 image: AssetImage("assets/twitter.png"),  
 height: 22.0,  
 ),  
 SizedBox(width: 10),  
 Text('Twitterでシェア',  
 style:TextStyle(  
 fontSize: 20,  
 color: Colors.*white*,  
 ),),  
 ],  
 ),  
 onPressed: () async {  
 await SocialSharePlugin.*shareToTwitterLink*(url: url);  
 },  
 color: Colors.*grey*,  
 ),  
 ],  
 ),  
 ),  
 actions: [  
 cancelButton,  
 ],  
 );  
  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return alert;  
 },  
 );  
 }  
  
}