Code for slide menu

Company side

Register and login

class LoginPage extends StatefulWidget {

  LoginPage({Key key, this.title, this.auth, this.onSignIn}) : super(key: key);

  final String title;

  final BaseAuth auth;

  final VoidCallback onSignIn;

  @override

  \_LoginPageState createState() => new \_LoginPageState();

}

enum FormType { login, register }

class \_LoginPageState extends State<LoginPage> with TickerProviderStateMixin {

  static final formKey = new GlobalKey<FormState>();

  final TextEditingController \_passwordTextController = TextEditingController();

  CrudMethods crudObj = new CrudMethods();

  String \_email;

  String \_password;

  DateTime dob;

  String \_name;

  FormType \_formType = FormType.login;

  String \_authHint = '';

  bool \_isLoading = false;

  bool userSex = true;

  int \_radioValue = 0;

  Color dobColor = Colors.grey[600];

  bool validateAndSave() {

    final form = formKey.currentState;

    if (form.validate()) {

      form.save();

      return true;

    }

    return false;

  }

  void validateAndSubmit() async {

    if (validateAndSave()) {

      setState(() {

        \_isLoading = true;

      });

      try {

        String userId = \_formType == FormType.login

            ? await widget.auth.signIn(\_email, \_password)

            : await widget.auth.createUser(\_email, \_password);

        setState(() {

          \_isLoading = false;

        });

        if (\_formType == FormType.register) {

          UserData userData = new UserData(

              name: \_name,

              surname: "",

              dob: dob,

              mail: \_email,

              notification: true,

              phone: "",

              picture:

                  "https://firebasestorage.googleapis.com/v0/b/lynight-53310.appspot.com/o/profilePics%2Fbloon\_pics.jpg?alt=media&token=ab6c1537-9b1c-4cb4-b9d6-2e5fa9c7cb46",

              reservation: [],

              sex: userSex,

                           friendRequest: [],

              friendList: [],);

          crudObj.createOrUpdateUserData(userData.getDataMap());

        }

        if (userId == null) {

          print("EMAIL NOT VERIFIED");

          setState(() {

            \_authHint = 'Check your email ;

            \_isLoading = false;

            \_formType = FormType.login;

          });

        } else {

          \_isLoading = false;

          widget.onSignIn();

        }

      } catch (e) {

        setState(() {

          \_isLoading = false;

          switch (e.code) {

            case 'ERROR\_INVALID\_EMAIL':

              \_authHint = 'Invalid email';

              break;

            case 'ERROR\_USER\_NOT\_FOUND':

              \_authHint = 'User cannot be found';

              break;

            case 'ERROR\_WRONG\_PASSWORD':

              \_authHint = 'Invalid password';

              break;

            default:

              \_authHint = 'Connection Error';

              break;

          }

        });

        print(e);

      }

    } else {

      setState(() {

        \_authHint = '';

      });

    }

  }

  void moveToRegister() {

    formKey.currentState.reset();

    setState(() {

      \_formType = FormType.register;

      \_authHint = '';

    });

  }

  void moveToLogin() {

    formKey.currentState.reset();

    setState(() {

      \_formType = FormType.login;

      \_authHint = '';

    });

  }

  Widget \_buildUsernameField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 50.0, 0.0, 0.0),

      child: TextFormField(

        maxLines: 1,

        key: new Key('email'),

        decoration: InputDecoration(

          labelText: 'Email',

          icon: new Icon(

            Icons.mail,

            color: Colors.grey,

          ),

        ),

        keyboardType: TextInputType.emailAddress,

        validator: (String value) {

          if (value.isEmpty ||

              !RegExp(r"[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*@(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?")

                  .hasMatch(value)) {

            return 'Enter a valid email';

          }

        },

        onSaved: (value) => \_email = value,

      ),

    );

  }

  Widget \_buildNameField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 15.0, 0.0, 0.0),

      child: TextFormField(

        maxLines: 1,

        key: new Key('namefield'),

        decoration: InputDecoration(

          labelText: 'First name',

          icon: new Icon(

            Icons.perm\_identity,

            color: Colors.grey,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Enter first name';

          }

        },

        onSaved: (value) => \_name = value,

      ),

    );

  }

  Widget \_buildPasswordField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 15.0, 0.0, 0.0),

      child: TextFormField(

        maxLines: 1,

        key: new Key('password'),

        decoration: InputDecoration(

            labelText: 'Password',

            icon: new Icon(

              Icons.lock,

              color: Colors.grey,

            )),

        controller: \_passwordTextController,

        obscureText: true,

        validator: (String value) {

          if (value.isEmpty || value.length < 6) {

            return 'Minimum of 6 characters required';

          }

        },

        onSaved: (value) => \_password = value,

      ),

    );

  }

  Widget \_builConfirmPasswordTextField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 15.0, 0.0, 0.0),

      child: TextFormField(

        decoration: InputDecoration(

            labelText: 'Confirm Password',

            icon: new Icon(

              Icons.lock,

              color: Colors.grey,

            )),

        obscureText: true,

        validator: (String value) {

          if (\_passwordTextController.text != value) {

            return 'Password mismatch';

          }

        },

      ),

    );

  }

  Widget \_showDatePicker() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 15.0, 0.0, 0.0),

      child: Row(

        children: <Widget>[

          Icon(

            Icons.date\_range,

            color: Colors.grey,

          ),

          FlatButton(

            onPressed: () {

              DatePicker.showDatePicker(

                context,

                showTitleActions: true,

                minTime: DateTime(1968, 1, 1),

                maxTime: DateTime(2030, 6, 7),

                onConfirm: (date) {

                  setState(() {

                    dob = date;

                    dobColor = Colors.grey[600];

                  });

                },

                currentTime: DateTime.now(),

                locale: LocaleType.fr,

              );

            },

            child: Text(

              dob == null

                  ? 'Date of Birth'

                  : DateFormat('dd/MM/yyyy').format(dob),

              style: TextStyle(color: dobColor, fontSize: 16.0),

            ),

          ),

        ],

      ),

    );

  }

  void \_handleRadioValueChange(int value) {

    setState(() {

      \_radioValue = value;

      switch (\_radioValue) {

        case 0:

          userSex = true;

          break;

        case 1:

          userSex = false;

          break;

      }

    });

  }

  Widget \_showGenderSelect() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(0.0, 3.0, 0.0, 0.0),

      child: Row(

        children: <Widget>[

          Icon(

            FontAwesomeIcons.venusMars,

            color: Colors.grey,

          ),

          Radio(

            value: 0,

            groupValue: \_radioValue,

            onChanged: \_handleRadioValueChange,

          ),

          Text(

            'Male',

            style: TextStyle(color: Colors.grey[600]),

          ),

          Radio(

            value: 1,

            groupValue: \_radioValue,

            onChanged: \_handleRadioValueChange,

          ),

          Text('Female', style: TextStyle(color: Colors.grey[600])),

        ],

      ),

    );

  }

  Widget submitWidgets() {

    switch (\_formType) {

      case FormType.login:

        return ListView(

          shrinkWrap: true,

          children: <Widget>[

            PrimaryButton(

              key: new Key('login'),

              text: 'Login',

              height: 44.0,

              onPressed: validateAndSubmit,

            ),

            FlatButton(

                key: new Key('need-account'),

                child: Text("Create Account"),

                onPressed: moveToRegister),

          ],

        );

      default:

        return ListView(

          shrinkWrap: true,

          children: <Widget>[

            PrimaryButton(

                key: new Key('register'),

                text: 'Register',

                height: 44.0,

                onPressed: () {

                  if (dob == null) {

                    validateAndSave();

                    setState(() {

                      dobColor = Colors.red[700];

                    });

                  } else {

                    validateAndSubmit();

                  }

                }),

            FlatButton(

                key: new Key('need-login'),

                child: Text("Already have an account? To log in"),

                onPressed: moveToLogin),

          ],

        );

    }

  }

  Widget \_showCircularProgress() {

    return Container(

      padding: const EdgeInsets.all(20.0),

      child: Center(child: CircularProgressIndicator()),

    );

  }

  Widget hintText() {

    return Container(

        //height: 80.0,

        padding: const EdgeInsets.all(32.0),

        child: Text(\_authHint,

            key: new Key('hint'),

            style: TextStyle(fontSize: 18.0, color: Colors.grey),

            textAlign: TextAlign.center));

  }

  Widget \_buildForm() {

    return Form(

        key: formKey,

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: <Widget>[

            // \_showLogo(),

            \_buildUsernameField(),

            \_formType == FormType.register ? \_buildNameField() : Container(),

            \_buildPasswordField(),

            \_formType == FormType.register

                ? \_builConfirmPasswordTextField()

                : Container(),

            \_formType == FormType.register ? \_showDatePicker() : Container(),

            SizedBox(

              height: 10.0,

            ),

            \_formType == FormType.register ? \_showGenderSelect() : Container(),

            \_isLoading == false ? submitWidgets() : \_showCircularProgress(),

//            \_formType != FormType.register ? googleSignIn() : Container(),

          ],

        ));

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(widget.title),

      ),

      backgroundColor: Colors.white,

      body: SingleChildScrollView(

        child: Container(

          padding: const EdgeInsets.all(16.0),

          child: Column(children: <Widget>[

            Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Container(

                padding: const EdgeInsets.all(16.0),

                child: \_buildForm(),

              ),

            ]),

            hintText(),

          ]),

        ),

      ),

    );

  }

  Widget padded({Widget child}) {

    return Padding(

      padding: EdgeInsets.symmetric(vertical: 8.0),

      child: child,

    );

  }

  Widget googleSignIn() {

    return OutlineButton(

      splashColor: Colors.grey,

      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(40)),

      highlightElevation: 0,

      borderSide: BorderSide(color: Colors.grey),

      child: Padding(

        padding: const EdgeInsets.fromLTRB(0, 10, 0, 10),

        child: Row(

          mainAxisSize: MainAxisSize.min,

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            Image(image: AssetImage("assets/google\_logo.png"), height: 35.0),

            Padding(

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

              child: Text(

                'Connect with Google',

                style: TextStyle(

                  fontSize: 20,

                  color: Colors.grey,

                ),

              ),

            )

          ],

        ),

      ),

      onPressed: (){

          widget.auth.signInWithGoogle();

      },

    );

  }

}

Paynow Payment with ecocash

class Homepage extends StatelessWidget {

  Homepage({this.documentID});

  final String documentID;

  // This widget is the root of your application.

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Paynow',

      theme: ThemeData(

        // This is the theme of your application.

        //

        // Try running your application with "flutter run". You'll see the

        // application has a blue toolbar. Then, without quitting the app, try

        // changing the primarySwatch below to Colors.green and then invoke

        // "hot reload" (press "r" in the console where you ran "flutter run",

        // or simply save your changes to "hot reload" in a Flutter IDE).

        // Notice that the counter didn't reset back to zero; the application

        // is not restarted.

      ),

      home: MyHomePage(title: 'Flutter Paynow '),

    );

  }

}

class MyHomePage extends StatefulWidget {

  MyHomePage({Key key, this.title}) : super(key: key);

  // This widget is the home page of your application. It is stateful, meaning

  // that it has a State object (defined below) that contains fields that affect

  // how it looks.

  // This class is the configuration for the state. It holds the values (in this

  // case the title) provided by the parent (in this case the App widget) and

  // used by the build method of the State. Fields in a Widget subclass are

  // always marked "final".

  final String title;

  @override

  \_MyHomePageState createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  TextEditingController \_phoneController = TextEditingController();

  TextEditingController \_amountController = TextEditingController();

  String \_status = "";

  List<GCircle> loaders = <GCircle>[];

  var parser = EmojiParser();

  @override

  void initState(){

    super.initState();

    // Timer.periodic(Duration(seconds: 5), (t){\_animateLoaders();});

  }

  @override

  Widget build(BuildContext context) {

    // This method is rerun every time setState is called, for instance as done

    // by the \_incrementCounter method above.

    //

    // The Flutter framework has been optimized to make rerunning build methods

    // fast, so that you can just rebuild anything that needs updating rather

    // than having to individually change instances of widgets.

    return Scaffold(

      backgroundColor: Colors.indigoAccent,

      appBar: AppBar(

        backgroundColor: Colors.greenAccent.withOpacity(.7),

      body:Stack(

        children: <Widget>[

          Padding(

            padding: EdgeInsets.only(right: 20, left: 20),

            child: Container(

              color: Colors.greenAccent.withOpacity(.5),

              child:Center(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Text(

                  'Ecocash Payment:',

                  style: TextStyle(

                    color: Colors.white

                  ),

                ),

                TextField(

                  decoration: InputDecoration(

                    icon: Icon(Icons.phone),

                    hintText: "Phone Number",

                    hintStyle: TextStyle(

                      color: Colors.white

                    )

                  ),

                  cursorColor:Colors.white ,

                  keyboardType: TextInputType.phone,

                  controller: \_phoneController,

                ),

                TextField(

                  decoration: InputDecoration(

                    hintText: "5.00",

                    hintStyle: TextStyle(color:Colors.white),

                    icon: Icon(Icons.monetization\_on)

                  ),

                  keyboardType: TextInputType.number,

                  controller: \_amountController,

                ),

                SizedBox(height: 15,),

                ArgonTimerButton(

                  height: 50,

                  width: MediaQuery.of(context).size.width \* 0.45,

                  minWidth: MediaQuery.of(context).size.width \* 0.30,

                  highlightColor: Colors.transparent,

                  highlightElevation: 0,

                  roundLoadingShape: false,

                  onTap: (startTimer, btnState) {

                    if (btnState == ButtonState.Idle) {

                      startTimer(15);

                    }

                    Paynow paynow = Paynow(integrationKey: "5a981990-d791-421e-b3e4-896c164b1e2e", integrationId: "6246", returnUrl: "http://google.com", resultUrl: "http://google.co");

                    Payment payment = paynow.createPayment("user", "user@email.com");

                    payment.add("Project", 1.9);

                    // Initiate Mobile Payment

                    paynow.sendMobile(payment, \_phoneController.text ?? "0785499644",)

                      ..then((InitResponse response){

                        // display results

                        print(response());

                        Future.delayed(Duration(seconds: 20~/2));

                        // Check Transaction status from pollUrl

                        paynow.checkTransactionStatus(response.pollUrl)

                          ..then((StatusResponse status){

                            print(status.paid);

                          });

                      });

                  },

                  child: Text(

                    "PAY",

                    style: TextStyle(

                        color: Colors.white,

                        fontSize: 18,

                        fontWeight: FontWeight.w700),

                  ),

                  loader: (timeLeft) {

                    return Text(

                      "Wait | $timeLeft",

                      style: TextStyle(

                          color: Colors.black,

                          fontSize: 18,

                          fontWeight: FontWeight.w700),

                    );

                  },

                  borderRadius: 5.0,

                  color: Colors.greenAccent ,

                  elevation: 0,

                  borderSide: BorderSide(color: Colors.black, width: 1.5),

                ),

                SizedBox(height: 10,),

              ],

            ),

          ))),

          Container(

            alignment: Alignment.centerLeft,

            child: Column(

              children: \_buildLoaders()

            ),

          ),

          Container(

            alignment: Alignment.centerRight,

            child: Column(

              children: \_buildLoaders()

            ),

          )

        ]

      ) ,

       // This trailing comma makes auto-formatting nicer for build methods.

    );

  }

  List<GCircle>\_buildLoaders() {

    this.loaders.clear();

    for(int i=0;i<3;i++){

      this.loaders.add(

        GCircle(Colors.black)

      );

    }

    return loaders;

  }

}

class GCircle extends StatefulWidget{

  Color color;

  GCircle(this.color);

  @override

  State<StatefulWidget> createState() {

    // TODO: implement createState

    return \_GCircle();

  }

  updateColor(Color color){

    this.color = color;

  }

}

class \_GCircle extends State<GCircle>{

Color c = Colors.black87;

  @override

  void initState(){

    List<Color> colors = [Colors.black , Colors.greenAccent, Colors.cyan];

    Random random = Random();

    Timer.periodic(Duration(seconds: 2), (t){

      setState(() {

        this.c = colors[random.nextInt(colors.length)];

      });

    });

    super.initState();

  }

  @override

  Widget build(BuildContext context) {

    // TODO: implement build

    return Container(

      child: Padding(

        padding: EdgeInsets.all(5),

        child: CircleAvatar(

          backgroundColor: c,

        ),

      )

    );

  }

}

Friends Page for payment

class FriendsPage extends StatefulWidget {

  FriendsPage({this.onSignOut});

//  final BaseAuth auth;

  final VoidCallback onSignOut;

  final BaseAuth auth = new Auth();

  void \_signOut() async {

    try {

      await auth.signOut();

      onSignOut();

    } catch (e) {

      print(e);

    }

  }

  @override

  State<StatefulWidget> createState() {

    // TODO: implement createState

    return \_FriendsPageState();

  }

}

class \_FriendsPageState extends State<FriendsPage> {

  String currentUserId = 'userId';

  String currentUserMail = 'userMail';

  Map<dynamic, dynamic> currentUserDataMap;

  String \_userName = '';

  List<Map<dynamic, dynamic>> listOfRequest = [];

  List<dynamic> userFriendRequestListFromFirestore = [];

  List<dynamic> userFriendList = [];

  List<Map<dynamic, dynamic>> friendListMap = [];

  CrudMethods crudObj = new CrudMethods();

  final SlidableController slidableController = SlidableController();

//  final GlobalKey<RefreshIndicatorState> \_refreshIndicatorKey = new GlobalKey<RefreshIndicatorState>();

  RefreshController \_refreshController;

  bool \_isLoading = false;

  @override

  void initState() {

    super.initState();

    \_refreshController = RefreshController(initialRefresh: true);

    widget.auth.currentUser().then((id) {

      setState(() {

        currentUserId = id;

      });

    });

    widget.auth.userEmail().then((mail) {

      setState(() {

        currentUserMail = mail;

      });

    });

  }

  Widget friendRequest() {

    List<dynamic> mutableListOfRequest = List.from(listOfRequest);

    if (mutableListOfRequest.isEmpty) {

      return SliverList(

        delegate: SliverChildListDelegate(

          [

            Center(

              child: Text('No Requests '),

            ),

          ],

        ),

      );

    }

    return SliverList(

      delegate: SliverChildBuilderDelegate(

            (BuildContext context, int i) {

          return Card(

            margin: EdgeInsets.fromLTRB(10.0, 0.0, 10.0, 20.0),

            child: Container(

              padding: EdgeInsets.fromLTRB(5.0, 10.0, 10.0, 10.0),

              child: Row(

                mainAxisAlignment: MainAxisAlignment.spaceBetween,

                children: <Widget>[

                  Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: <Widget>[

                      CircleAvatar(

                        // photo of profile

                        backgroundImage:

                        NetworkImage(mutableListOfRequest[i]['picture']),

                        minRadius: 25,

                        maxRadius: 25,

                      ),

                      SizedBox(

                        width: 10,

                      ),

                      Container(

                        width: 200,

                        child: Column(

                          mainAxisAlignment: MainAxisAlignment.start,

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: <Widget>[

                            Text(

                              mutableListOfRequest[i]['name'],

                              style: TextStyle(

                                  fontSize: 16.0, fontWeight: FontWeight.w400),

                            ),

                            Text(

                              mutableListOfRequest[i]['mail'],

                              style: TextStyle(

                                  color: Colors.grey[600], fontSize: 12.0),

                            ),

                          ],

                        ),

                      ),

                    ],

                  ),

                  InkWell(

                    //add

                    child: Icon(FontAwesomeIcons.check,color: Colors.green[400],),

                    onTap: () {

                      userFriendRequestListFromFirestore =

                      List.from(userFriendRequestListFromFirestore)

                        ..removeAt(i);

                      addFriend(mutableListOfRequest[i]['ID'],

                          mutableListOfRequest[i]['friendList']);

                      setState(() {

                        listOfRequest = List.from(listOfRequest)

                          ..removeAt(i);

                      });

                    },

                  ),

                  InkWell(

                    //remove

                    child: Icon(FontAwesomeIcons.times,color: Colors.red[400],),

                    onTap: () {

                      userFriendRequestListFromFirestore =

                      List.from(userFriendRequestListFromFirestore)

                        ..removeAt(i);

                      removeFriendRequest();

                      setState(() {

                        listOfRequest = List.from(listOfRequest)

                          ..removeAt(i);

                      });

                    },

                  ),

                ],

              ),

            ),

          );

        },

        childCount: mutableListOfRequest.length,

      ),

    );

  }

  void removeFriendRequest() {

    crudObj.updateData('user', currentUserId,

        {'friendRequest': userFriendRequestListFromFirestore});

  }

  void addFriend(iDDemander, friendListOfDemander) {

    if (friendListOfDemander != null) {

      List<dynamic> mutableFriendListOfDemander =

      List.from(friendListOfDemander);

      mutableFriendListOfDemander.add(currentUserId);

      crudObj.updateData('user', iDDemander, {

        'friendList': mutableFriendListOfDemander

      });

    } else {

      crudObj.updateData('user', iDDemander, {

        'friendList': [currentUserId]

      });

    }

    if (currentUserDataMap['friendList'] != null) {

      List<dynamic> mutableFriendListOfCurrentUser =

      List.from(currentUserDataMap['friendList']);

      mutableFriendListOfCurrentUser.add(iDDemander);

      crudObj.updateData('user', currentUserId, {

        'friendList': mutableFriendListOfCurrentUser

      });

      crudObj.updateData('user', currentUserId,

          {'friendRequest': userFriendRequestListFromFirestore});

    } else {

      crudObj.updateData('user', currentUserId, {

        'friendList': [iDDemander]

      });

      crudObj.updateData('user', currentUserId,

          {'friendRequest': userFriendRequestListFromFirestore});

    }

    \_refresh();

  }

  Widget friendList() {

    if (friendListMap.isEmpty) {

      return SliverList(

        delegate: SliverChildListDelegate(

          [

            Center(

              child: Text('No friends'),

            ),

          ],

        ),

      );

    }

    return SliverList(

      delegate: SliverChildBuilderDelegate((BuildContext context, int i) {

        return Column(

          children: <Widget>[

            Slidable(

              controller: slidableController,

              key: Key(Random().nextInt(1000).toString() +

                  friendListMap[i]['name'].toString()),

              actionPane: SlidableScrollActionPane(),

              actionExtentRatio: 0.25,

              actions: <Widget>[

                \_inviteFriendSlideButton(i),

              ],

              secondaryActions: <Widget>[

                \_deleteFriendSlideButton(i),

              ],

              child: \_makeCard(i),

            ),

            i == friendListMap.length - 1 ? SizedBox(height: 200.0,) : Container(),

          ],

        );

      }, childCount: friendListMap.length),

    );

  }

  Widget \_makeCard(i) {

    return Card(

      color: Colors.transparent,

      elevation: 0.0,

      margin: EdgeInsets.symmetric(horizontal: 10.0, vertical: 6.0),

      child: Container(

        decoration: BoxDecoration(

          gradient: LinearGradient(

              begin: Alignment.topLeft,

              end: Alignment.bottomRight,

              colors: [

                Color.fromRGBO(212, 63, 141, 1),

                Color.fromRGBO(2, 80, 197, 1)

              ]),

          //color: Theme.of(context).primaryColor,

          borderRadius: BorderRadius.all(

            Radius.circular(25),

          ),

        ),

        child: \_makeListTile(i),

      ),

    );

  }

  Widget \_makeListTile(i) {

    return ListTile(

      contentPadding: EdgeInsets.symmetric(horizontal: 20.0, vertical: 10.0),

      leading: CircleAvatar(

        backgroundImage: NetworkImage(friendListMap[i]['picture']),

        minRadius: 25,

        maxRadius: 25,

      ),

      title: Text(

        friendListMap[i]['name'],

        style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),

      ),

      subtitle: Text(

          friendListMap[i]['mail'], style: TextStyle(color: Colors.white)),

          trailing:

          Icon(Icons.keyboard\_arrow\_right, color: Colors.white, size: 25.0),

      onTap: () {

        Navigator.push(

          context,

          MaterialPageRoute(

                    builder: (context) => Homepage(

                          documentID:  friendListMap[i]['ID'],

                        )));

      },

    );

  }

  Widget \_deleteFriendSlideButton(i) {

    return IconSlideAction(

      caption: 'Remove',

      color: Colors.red,

      icon: Icons.delete,

      onTap: () {

        List<dynamic> mutableFriendList = List.from(userFriendList);

        List<Map<dynamic, dynamic>> mutableFriendListMap =

            friendListMap;

        String friendID = friendListMap[i]['ID'];

//

        for (int b = 0; b < mutableFriendList.length; b++) {

          if (friendID == mutableFriendList[b]) {

            mutableFriendList.removeAt(b);

          }

        }

        List<dynamic> mutableFriendListOfFriend =

        List.from(friendListMap[i]['friendList']);

        for (int k = 0; k < mutableFriendListOfFriend.length; k++) {

          if (currentUserId == mutableFriendListOfFriend[k]) {

            mutableFriendListOfFriend.removeAt(k);

            break;

          }

        }

        mutableFriendListMap = List.from(mutableFriendListMap)

          ..removeAt(i);

        setState(() {

          userFriendList = mutableFriendList;

          friendListMap = mutableFriendListMap;

        });

        removeFriend(friendID, mutableFriendListOfFriend);

      },

    );

  }

  Widget \_inviteFriendSlideButton(i) {

    return IconSlideAction(

      caption: 'Invite',

      color: Colors.blue,

      icon: FontAwesomeIcons.handshake,

      onTap: () {

        \_openModalBottomSheet(

            context, friendListMap[i]['ID'], friendListMap[i]['name']);

      },

    );

  }

  void removeFriend(friendID, friendListOfFriend) {

    crudObj.updateData('user', currentUserId, {'friendList': userFriendList});

    crudObj.updateData('user', friendID, {'friendList': friendListOfFriend});

    \_refresh();

  }

  void \_openModalBottomSheet(context, friendID, friendName) {

    showModalBottomSheet(

        context: context,

        builder: (context) {

          return Scaffold(

            body: Container(

              decoration: BoxDecoration(

                border: Border.all(color: Color(0xFF737373)),

                color: Color(0xFF737373),

              ),

              child: Container(

                child: inviteToClub(friendID, friendName),

                decoration: BoxDecoration(

                  color: Colors.white,

                  borderRadius: BorderRadius.only(

                    topLeft: const Radius.circular(25.0),

                    topRight: const Radius.circular(25.0),

                  ),

                ),

              ),

            ),

          );

        });

  }

Add Event class

class \_AddEventState extends State<AddClub> {

  String userMail = 'userMail';

  String userId = 'userId';

  static final formKey = new GlobalKey<FormState>();

  CrudMethods crudObj = new CrudMethods();

  bool \_isLoading = false;

  String \_name;

  String \_description;

  String \_adress;

  String \_phone;

  String \_siteUrl;

  double \_latitude;

  double \_longitude;

  @override

  void initState() {

    super.initState();

    widget.auth.currentUser().then((id) {

      setState(() {

        userId = id;

      });

    });

    widget.auth.userEmail().then((mail) {

      setState(() {

        userMail = mail;

      });

    });

  }

  void validateAndSubmit() async {

    if (validateAndSave()) {

      setState(() {

        \_isLoading = true;

      });

      EventData eventData = new EventData(

        name: \_name,

        description: \_description,

        adress: \_adress,

        phone: \_phone,

        siteUrl: \_siteUrl,

        position: GeoPoint(\_latitude, \_longitude),

        storagePath: '',

        searchKey: \_name.substring(0, 1).toUpperCase(),

      );

 DocumentReference docRef = await Firestore.instance

          .collection('event')

          .add(eventData.getClubDataMap());

  Widget submitWidget() {

    return PrimaryButton(

        key: new Key('submitEvent'),

        text: 'Create',

        height: 44.0,

        onPressed: () {

            validateAndSubmit();

        });

  }

  Widget \_eventNameField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 0.0),

      child: TextFormField(

        maxLines: 1,

        key: new Key('eventName'),

        decoration: InputDecoration(

          labelText: 'Event Name',

          icon: new Icon(

            FontAwesomeIcons.solidMoon,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Enter event name';

          }

        },

        onSaved: (value) => \_name = value,

      ),

    );

  }

  Widget \_eventDescription() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 30.0, 10.0, 15.0),

      child: TextFormField(

        maxLength: 200,

        key: new Key('eventDescription'),

        decoration: InputDecoration(

          labelText: 'Event Decription',

          icon: new Icon(

            FontAwesomeIcons.solidClipboard,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Enter Event Description';

          }

        },

        onSaved: (value) => \_description = value,

      ),

    );

  }

  Widget \_eventAdress() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 0.0),

      child: TextFormField(

        key: new Key('eventAdress'),

        decoration: InputDecoration(

          labelText: 'EventAddress',

          icon: new Icon(

            FontAwesomeIcons.mapMarkerAlt,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Enter Event Address';

          }

        },

        onSaved: (value) => \_adress = value,

      ),

    );

  }

  Widget \_eventPhone() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 0.0),

      child: TextFormField(

        key: new Key('eventContact'),

        keyboardType: TextInputType.number,

        decoration: InputDecoration(

          labelText: 'Event Contact',

          icon: new Icon(

            FontAwesomeIcons.phone,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Enter the phone number';

          }

        },

        onSaved: (value) => \_phone = value,

      ),

    );

  }

  Widget \_eventPosition() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 50.0, 10.0, 25.0),

      child: Column(

        children: <Widget>[

          TextFormField(

            key: new Key('eventPositionLat'),

            keyboardType: TextInputType.number,

            decoration: InputDecoration(

              labelText: 'Latitude',

              icon: new Icon(

                FontAwesomeIcons.mapPin,

                size: 24,

                color: Theme.of(context).primaryColor,

              ),

            ),

            validator: (String value) {

              if (value.isEmpty) {

                return 'Enter the latitude';

              }

            },

            onSaved: (value) => \_latitude = double.parse(value),

          ),

          TextFormField(

            key: new Key('eventPositionLon'),

            keyboardType: TextInputType.number,

            decoration: InputDecoration(

              labelText: 'Longitude',

              icon: new Icon(

                FontAwesomeIcons.mapPin,

                size: 24,

                color: Theme.of(context).primaryColor,

              ),

            ),

            validator: (String value) {

              if (value.isEmpty) {

                return 'Enter the longitude';

              }

            },

            onSaved: (value) => \_longitude = double.parse(value),

          ),

        ],

      ),

    );

  }

  Widget \_eventSiteUrl() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 25.0),

      child: TextFormField(

        key: new Key('eventUrl'),

        decoration: InputDecoration(

          labelText: 'Site web [optional]',

          icon: new Icon(

            FontAwesomeIcons.link,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        //pas de validator car le site web est optionnel

//        validator: (String value) {

//          if (value.isEmpty) {

//            return 'Saisissez un numéro';

//          }

//        },

        onSaved: (value) => \_siteUrl = value,

      ),

    );

  }

  bool validateAndSave() {

    final form = formKey.currentState;

    if (form.validate()) {

      form.save();

      return true;

    }

    return false;

  }

  Widget \_showCircularProgress() {

    return Container(

      padding: const EdgeInsets.all(20.0),

      child: Center(child: CircularProgressIndicator()),

    );

  }

  Widget buildForm() {

    return Form(

      key: formKey,

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.stretch,

        children: <Widget>[

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

            child: Column(

              children: <Widget>[

                \_eventNameField(),

                \_eventDescription(),

              ],

            ),

          ),

          Container(

            height: 30,

          ),

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

            child: Column(

              children: <Widget>[

                \_eventPhone(),

                \_eventAdress(),

                \_eventPosition(),

              ],

            ),

          ),

          Container(

            height: 30,

          ),

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

            child: Column(

              children: <Widget>[

                \_eventSiteUrl(),

              ],

            ),

          ),

          \_isLoading == false ? submitWidget() : \_showCircularProgress(),

        ],

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    // TODO: implement build

    return Scaffold(

      backgroundColor: Colors.white,

      appBar: AppBar(

        iconTheme: IconThemeData(color: Theme.of(context).primaryColor),

        backgroundColor: Colors.white,

        title: Text(

          'Add Event',

          style: TextStyle(color: Theme.of(context).primaryColor, fontSize: 30),

        ),

      ),

      body: SingleChildScrollView(

        child: Container(

          padding: const EdgeInsets.all(16.0),

          child: Column(children: <Widget>[

            Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Container(

                padding: const EdgeInsets.all(16.0),

                child: buildForm(),

              ),

            ]),

          ]),

        ),

      ),

      drawer: CustomSlider(

        userMail: userMail,

        signOut: widget.\_signOut,

        activePage: 'AddEvent',

      ),

    );

  }

}

Client Side

Map View of events

Class GoogleMapsClient extends StatefulWidget {

  GoogleMapsClient({this.onSignOut});

  final VoidCallback onSignOut;

  final BaseAuth auth = new Auth();

  void \_signOut() async {

    try {

      await auth.signOut();

      onSignOut();

    } catch (e) {

      print(e);

    }

  }

  @override

  State<StatefulWidget> createState() {

    // TODO: implement createState

    return \_GoogleMapsState();

  }

}

class \_GoogleMapsState extends State<GoogleMapsClient> {

  Completer<GoogleMapController> \_controller = Completer();

  static LatLng \_center = LatLng(-17.8395, 31.0075);

  final Set<Marker> \_markers = {};

  Location location = new Location();

  String userMail = 'userMail';

  @override

  void initState() {

    super.initState();

    widget.auth.userEmail().then((mail) {

      setState(() {

        userMail = mail;

      });

    });

  }

  void placeAllMarkers() async {

    QuerySnapshot snapshot =

        await Firestore.instance.collection('event').getDocuments();

    for (int i = 0; i < snapshot.documents.length; i++) {

      \_addMarkers(

          snapshot.documents[i]['position'].latitude,

          snapshot.documents[i]['position'].longitude,

          snapshot.documents[i]['name'],

          snapshot.documents[i]['description'],

          snapshot.documents[i].documentID);

          print( snapshot.documents[i].documentID);

    }

  }

  void \_addMarkers(latitude, longitude, eventName, eventDesc, eventID) {

    LatLng \_nightClubPosition = LatLng(latitude, longitude);

    setState(() {

      \_markers.add(Marker(

        markerId: MarkerId(\_nightClubPosition.toString()),

        position: \_nightClubPosition,

        infoWindow: InfoWindow(

          title: eventName,

          snippet: eventDesc,

          onTap: () {

            Navigator.push(

                context,

                MaterialPageRoute(

                    builder: (context) => NightClubProfile(

                          documentID: eventID,

                        )));

          },

        ),

        icon: BitmapDescriptor.fromAsset('assets/bloon\_final\_pin.png'),

      ));

    });

  }

  Future<void> \_userPosition() async {

    GoogleMapController mapController = await \_controller.future;

    var pos = await location.getLocation();

    mapController.animateCamera(CameraUpdate.newCameraPosition(CameraPosition(

      target: LatLng(pos.latitude, pos.longitude),

      zoom: 18.0,

    )));

  }

  void \_onMapCreated(GoogleMapController controller) {

    location.requestPermission();

    \_controller.complete(controller);

  }

  Widget googleMapsCreation(){

    return GoogleMap(

      initialCameraPosition: CameraPosition(

        target: \_center,

        zoom: 13,

      ),

      onMapCreated: \_onMapCreated,

      myLocationEnabled: true,

      rotateGesturesEnabled: true,

      zoomGesturesEnabled: true,

      scrollGesturesEnabled: true,

      markers: \_markers,

    );

  }

  @override

  Widget build(BuildContext context) {

    placeAllMarkers();

    // TODO: implement build

    return Scaffold(

      resizeToAvoidBottomPadding: false,

      appBar: AppBar(

        iconTheme: IconThemeData(color: Theme.of(context).primaryColor),

        title: Text(

          'View On Map',

          style: TextStyle(color: Theme.of(context).primaryColor, fontSize: 30),

        ),

        backgroundColor: Colors.white,

      ),

      body: Container(

        child: Stack(children: <Widget>[

          googleMapsCreation(),

          Align(

            alignment: Alignment.topRight,             child: Column(

              children: <Widget>[

                SizedBox(height: 3),

                FloatingActionButton(

                  onPressed: \_userPosition,

                  backgroundColor: Colors.white,

                  child: const Icon(

                    Icons.center\_focus\_weak,

                    size: 35,

                    color: Color(0xFF7854d3),

                  ),

                ),

                SizedBox(height: 30),

//

              ],

            ),

          ),

        ]),

        constraints: BoxConstraints(

            maxHeight: double.infinity, maxWidth: double.infinity),

      ),

      backgroundColor: Theme.of(context).accentColor,

      drawer: CustomSlider(

        userMail: userMail,

        signOut: widget.\_signOut,

        activePage: 'Maps',

      ),

    );

  }

}

Add Project Progress

class AddClub extends StatefulWidget {

  AddClub({this.onSignOut});

  final VoidCallback onSignOut;

  final BaseAuth auth = new Auth();

  void \_signOut() async {

    try {

      await auth.signOut();

      onSignOut();

    } catch (e) {

      print(e);

    }

  }

  @override

  State<StatefulWidget> createState() {

    // TODO: implement createState

    return \_AddClubState();

  }

}

class \_AddClubState extends State<AddClub> {

  String userMail = 'userMail';

  String userId = 'userId';

  static final formKey = new GlobalKey<FormState>();

  CrudMethods crudObj = new CrudMethods();

  bool \_isLoading = false;

  String \_name;

  String \_description;

  String \_siteUrl;

  List<File> clubPictureFile = new List<File>(4);

  bool idealisation = false;

  bool development = false;

  bool prototyping = false;

  bool marketing= false;

  @override

  void initState() {

    super.initState();

    widget.auth.currentUser().then((id) {

      setState(() {

        userId = id;

      });

    });

    widget.auth.userEmail().then((mail) {

      setState(() {

        userMail = mail;

      });

    });

  }

  void validateAndSubmit() async {

    if (validateAndSave()) {

      setState(() {

        \_isLoading = true;

      });

      ProjectData clubData = new ProjectData(

        name: \_name,

        description: \_description,

        siteUrl: \_siteUrl,

        stages: {

          'idealisation': idealisation,

          'development': development,

          'prototyping': prototyping,

          'Marketing': marketing,

        },

        pictures: [],

      );

      DocumentReference docRef = await Firestore.instance

          .collection('project')

          .add(clubData.getProjectDataMap());

      uploadPictures(docRef.documentID);

     setState(() {

       \_isLoading = false;

     });

    }

  }

  Widget submitWidget() {

    return PrimaryButton(

        key: new Key('submitproject'),

        text: 'Submit',

        height: 44.0,

        onPressed: () {

          if (clubPictureFile[0] == null) {

            validateAndSave();

            \_showDialogMissingPhoto();

          } else if (!idealisation &&

              !development&&

              !prototyping &&

              !marketing

              ) {

            validateAndSave();

            \_showDialogMissingMusicStyle();

          } else {

            validateAndSubmit();

          }

        });

  }

  void \_showDialogMissingPhoto() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: new Text("Project Photo"),

          content: new Text(

            "Please add atleast 1 picture",

            textAlign: TextAlign.justify,

          ),

          actions: <Widget>[

            new FlatButton(

              child: new Text("Ok"),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  void \_showDialogMissingMusicStyle() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: new Text("Project Stage"),

          content: new Text(

            "Please select atleast 1 Stage",

            textAlign: TextAlign.justify,

          ),

          actions: <Widget>[

            new FlatButton(

              child: new Text("Ok"),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  Widget \_clubNameField() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 0.0),

      child: TextFormField(

        maxLines: 1,

        key: new Key('projectName'),

        decoration: InputDecoration(

          labelText: 'Project Name',

          icon: new Icon(

            FontAwesomeIcons.solidMoon,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'Project Name should not be empty';

          }

        },

        onSaved: (value) => \_name = value,

      ),

    );

  }

  Widget \_clubDescription() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 30.0, 10.0, 15.0),

      child: TextFormField(

        maxLength: 200,

        key: new Key(' projectDescription'),

        decoration: InputDecoration(

          labelText: 'Decription',

          icon: new Icon(

            FontAwesomeIcons.solidClipboard,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        validator: (String value) {

          if (value.isEmpty) {

            return 'description should not be empty';

          }

        },

        onSaved: (value) => \_description = value,

      ),

    );

  }

  Widget checkbox(String title, bool boolValue) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: <Widget>[

        Text(

          title,

          style: TextStyle(color: Colors.grey[600]),

        ),

        Checkbox(

          value: boolValue,

          onChanged: (bool value) {

            switch (title) {

              case "Idealisation":

                setState(() {

                  idealisation = value;

                });

                break;

              case "Development":

                setState(() {

                  development = value;

                });

                break;

              case "Prototyping":

                setState(() {

                  prototyping = value;

                });

                break;

              case "Marketing":

                setState(() {

                 marketing = value;

                });

                break;

            }

          },

        )

      ],

    );

  }

  Widget \_checkboxMusicStyle() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 30.0, 10.0, 0.0),

      child: Column(

        children: <Widget>[

          Center(

            child: Text(

              'Project Development Stages',

              style: TextStyle(

                  fontSize: 20, color: Theme.of(context).primaryColor),

            ),

          ),

          SizedBox(

            height: 15,

          ),

          Row(

            mainAxisAlignment: MainAxisAlignment.spaceEvenly,

            children: <Widget>[

              checkbox('Idealisation', idealisation),

              SizedBox(

                width: 20,

              ),

              checkbox('Development', development),

            ],

          ),

          Row(

            mainAxisAlignment: MainAxisAlignment.spaceEvenly,

            children: <Widget>[

              checkbox('Prototyping', prototyping),

              SizedBox(

                width: 20,

              ),

              checkbox('Marketing', marketing),

            ],

          ),

        ],

      ),

    );

  }

  Widget \_clubSiteUrl() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 20.0, 10.0, 25.0),

      child: TextFormField(

        key: new Key(' projectUrl'),

        decoration: InputDecoration(

          labelText: 'Site web [optionnal]',

          icon: new Icon(

            FontAwesomeIcons.link,

            size: 24,

            color: Theme.of(context).primaryColor,

          ),

        ),

        onSaved: (value) => \_siteUrl = value,

      ),

    );

  }

  Widget \_selectionPictures() {

    return Padding(

      padding: const EdgeInsets.fromLTRB(5.0, 50.0, 10.0, 0.0),

      child: Container(

        height: 220,

        child: ListView.builder(

          scrollDirection: Axis.horizontal,

          shrinkWrap: true,

          itemCount: clubPictureFile.length,

          itemBuilder: (BuildContext context, int index) {

            return Column(

              key: Key('pic$index'),

              children: <Widget>[

                Container(

                  decoration: BoxDecoration(

                      borderRadius: BorderRadius.circular(20.0),

                      color: Colors.grey[300]),

                  margin: EdgeInsets.only(right: 10),

                  width: 250,

                  height: 200,

                  child: clubPictureFile[index] == null

                      ? FlatButton(

                          onPressed: () {

                            getImageFromGallery(index);

                          },

                          child: Icon(Icons.add\_circle\_outline),

                        )

                      : InkWell(

                          onTap: () {

                            getImageFromGallery(index);

                          },

                          child: Image.file(

                            clubPictureFile[index],

                            height: 200,

                            width: 200,

                            fit: BoxFit.cover,

                          ),

                        ),

                ),

                Text(

                  'Photo ${index + 1}',

                  style: TextStyle(color: Colors.grey[600]),

                ),

              ],

            );

          },

        ),

      ),

    );

  }

  Future getImageFromGallery(picNumber) async {

    var tempImage = await ImagePicker.pickImage(source: ImageSource.gallery);

    if (picNumber == 0) {

      setState(() {

        projectPictureFile[0] = tempImage;

      });

    }

    if (picNumber == 1) {

      setState(() {

         projectPictureFile[1] = tempImage;

      });

    }

    if (picNumber == 2) {

      setState(() {

        projectPictureFile[2] = tempImage;

      });

    }

    if (picNumber == 3) {

      setState(() {

         projectPictureFile[3] = tempImage;

      });

    }

  }

  uploadPictures( projectID) async {

//    FirebaseUser user = await FirebaseAuth.instance.currentUser();

//    var rnd = new Random();

    List<String> urlList = [];

    for (int i = 0; i <  projectPictureFile.length; i++) {

      if  projectPictureFile[i] != null) {

        final StorageReference firebaseStorageRef =

            FirebaseStorage.instance.ref().child('projectPics/$clubID/$i.jpg');

        final StorageUploadTask task =

            firebaseStorageRef.putFile( projectPictureFile[i]);

//      if (task.isInProgress) {

//        setState(() {

//          \_isLoading = true;

//        });

//      }

        var downloadUrl = await (await task.onComplete).ref.getDownloadURL();

        var url = downloadUrl.toString();

        urlList.add(url);

      }

    }

    setState(() {

      \_isLoading = false;

    });

    updateProjectPictures(urlList,projectID);

  }

  updateprojectPictures(picUrlList, projectID) {

    Firestore.instance

        .collection('project')

        .document(projectID)

        .updateData({"pictures": picUrlList});

    Navigator.pushReplacementNamed(context,

        "/");

  bool validateAndSave() {

    final form = formKey.currentState;

    if (form.validate()) {

      form.save();

      return true;

    }

    return false;

  }

  Widget \_showCircularProgress() {

    return Container(

      padding: const EdgeInsets.all(20.0),

      child: Center(child: CircularProgressIndicator()),

    );

  }

  Widget buildForm() {

    return Form(

      key: formKey,

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.stretch,

        children: <Widget>[

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

            child: Column(

              children: <Widget>[

                \_clubNameField(),

                \_clubDescription(),

              ],

            ),

          ),

          Container(

            height: 30,

          ),

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

          Container(

            height: 30,

          ),

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

          ),

          \_selectionPictures(),

          Container(

            height: 30,

          ),

          Container(

            width: MediaQuery.of(context).size.width,

            decoration: BoxDecoration(

              color: Colors.white,

              borderRadius: BorderRadius.all(Radius.circular(20.0)),

              boxShadow: [

                BoxShadow(

                  color: Colors.grey,

                  offset: Offset(2.0, 5.0),

                  blurRadius: 10.0,

                )

              ],

            ),

            child: Column(

              children: <Widget>[

                \_checkboxMusicStyle(),

                \_clubSiteUrl(),

              ],

            ),

          ),

          \_isLoading == false ? submitWidget() : \_showCircularProgress(),

        ],

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    // TODO: implement build

    return Scaffold(

      backgroundColor: Colors.white,

      appBar: AppBar(

        iconTheme: IconThemeData(color: Theme.of(context).primaryColor),

        backgroundColor: Colors.white,

        title: Text(

          'Add Progress',

          style: TextStyle(color: Theme.of(context).primaryColor, fontSize: 30),

        ),

      ),

      body: SingleChildScrollView(

        child: Container(

          padding: const EdgeInsets.all(16.0),

          child: Column(children: <Widget>[

            Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Container(

                padding: const EdgeInsets.all(16.0),

                child: buildForm(),

              ),

            ]),

          ]),

        ),

      ),

      drawer: CustomSlider(

        userMail: userMail,

        signOut: widget.\_signOut,

        activePage: 'AddEvent',

      ),

    );

  }

}