Learn a Fruit - Flutter Application

Current Trends in Software Engineering SE-4010

Final Project Report

Submitted by:

W.M.H.B. Warnakulasooriya
IT17136402

Table of Contents

1. Introduction

1.1 Description of The Application

This flutter application is used to learn about fruits. A splash screen with an animation gives the feeling of a perfect application to the user. Home screen of this app consists with a beautiful carousel slider in order to give the user the idea about what this app does. Neatly laid out and user-friendly interfaces are used in here. The color settings and perfect UI layouts lead the user to explore the application with zero user ambiguity.

User can add fruits as they like to their fruit collection. User can add an image of the fruit directly from granting the camera access or from the local file storage. User can add the fruit name, fruit family and the countries which that fruit is available. This is called the Fruit Book. They can view the added fruits later. They can edit the added details later. Or else they can delete them if those fruits are not needed for the user anymore.

User can maintain a favorite fruit collection also. User can add fruits from their fruit collection to favorite fruit collection. This is the general functionality of adding a bookmark. Later if the user wishes to remove some fruits from their favorite fruit collection, they can remove them too.

User can manage their profile. They can change their user details if they wish. They can add a profile picture for their account. They can change it if they wish later, or even they can remove the picture.

About us page is all about the people who developed the app. Our details have been added in this about us page.

2. Description about Screens

2.1 Splash Screen

Learn a Fruit app has an animated splash screen which is an added future for this app.

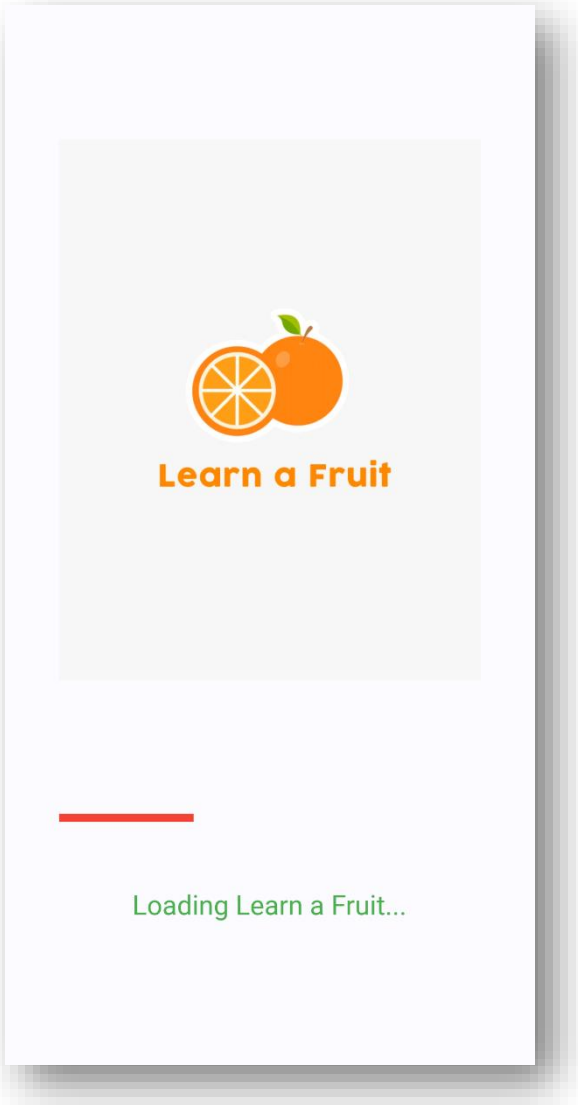

Figure 1: Splash screen of Learn a Fruit

2.2 Sign Up and Login Screens

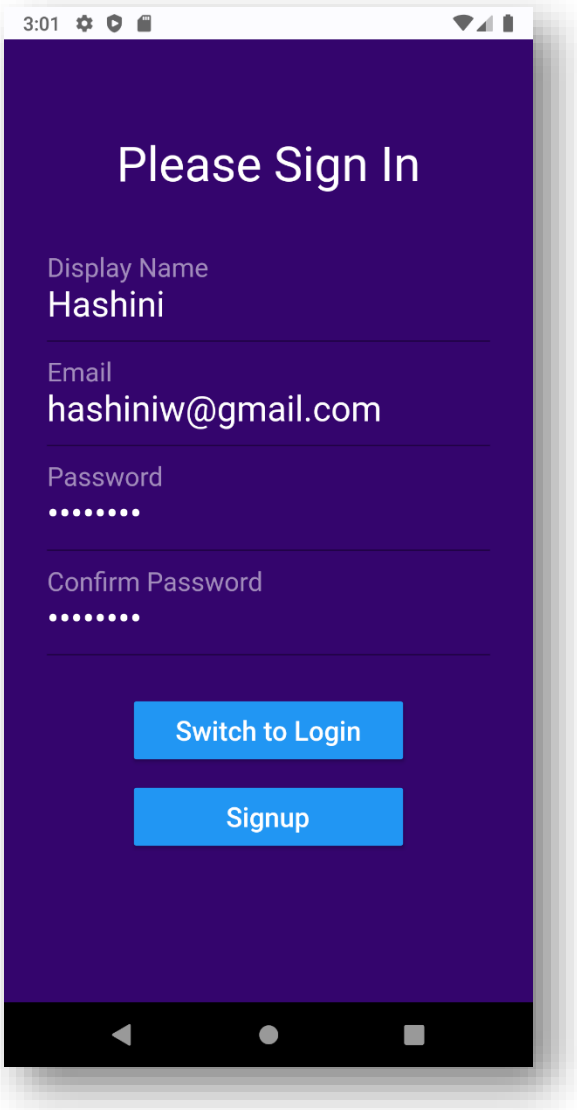

Figure 2: Sign-Up page

This is the user sign up page. Here if the user enters invalid inputs, the correct validations will be done. The user will be able to successfully become a member by filling out valid information in this page and signing up.

If we go to the database and look at the authentication tab, we can see that this email has been successfully added. That means, our flutter app and firebase is perfectly connected.

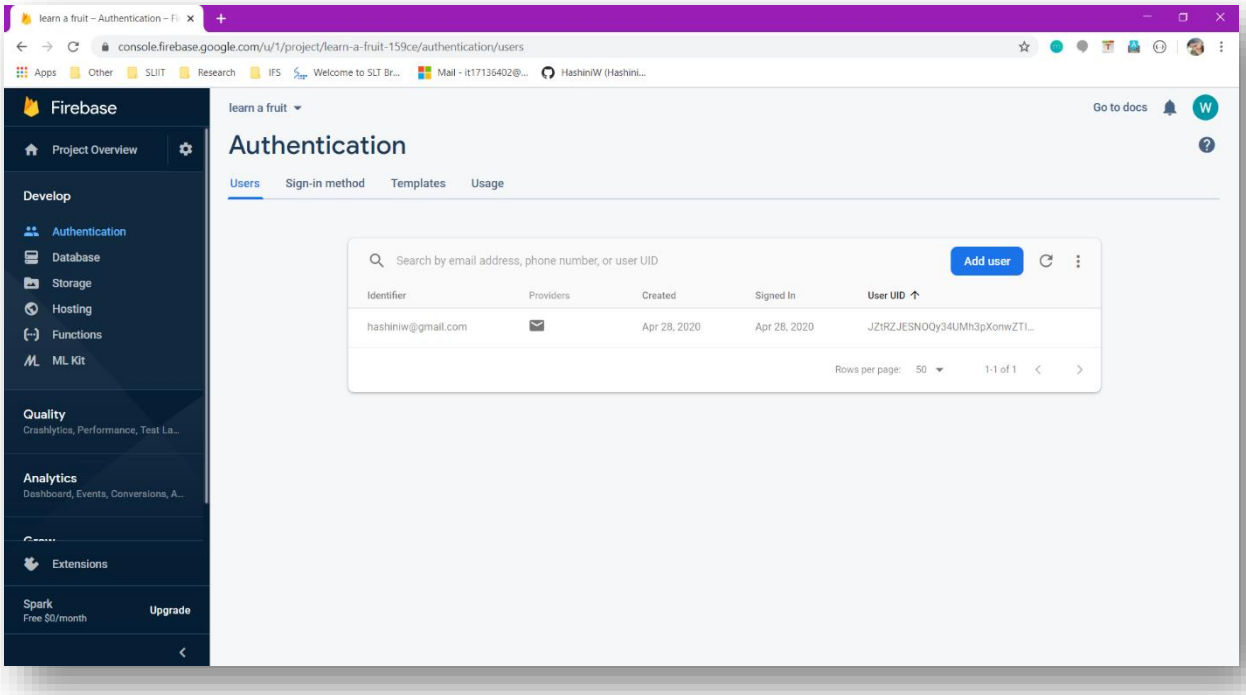

Figure 2.1: User authentication in Firebase

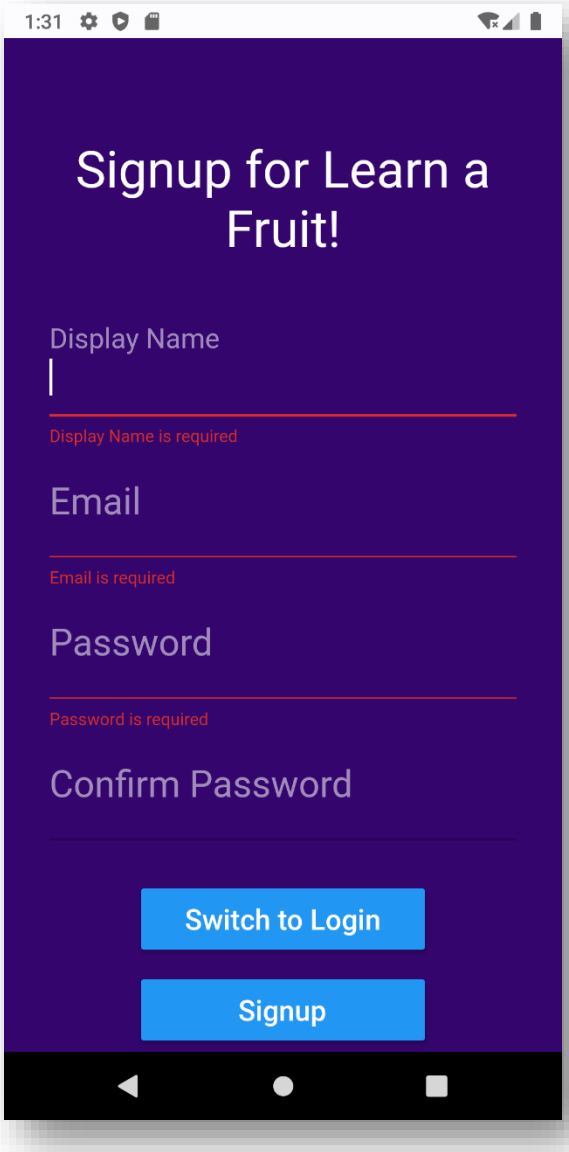

Figure 2.2: Validated fields in the Sign-Up page

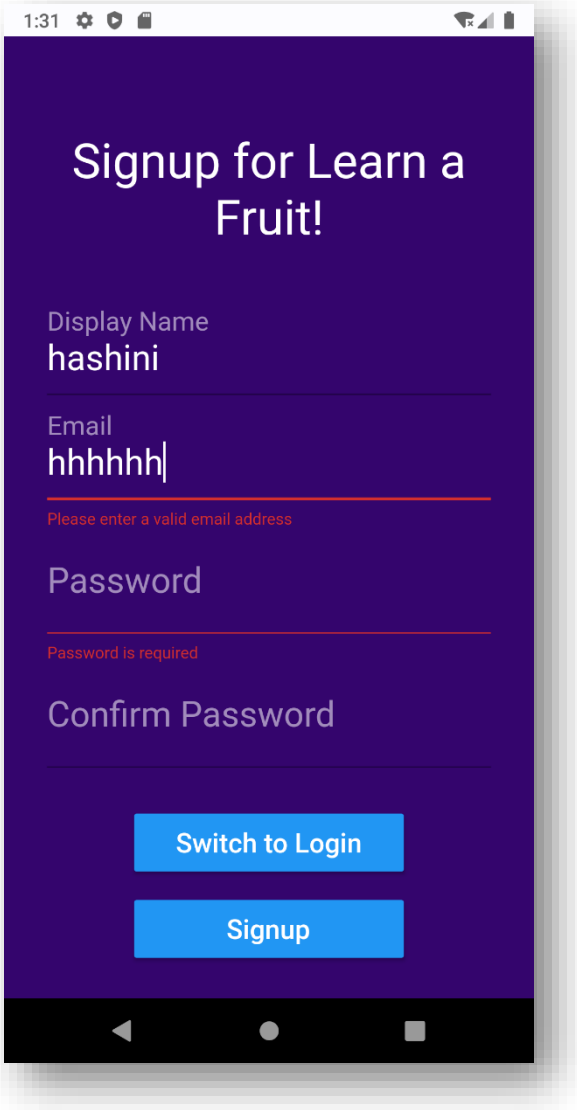

Figure 2.3: Validation called when invalid email is being typed

Email must be entered in the correct format in order to successfully signup.

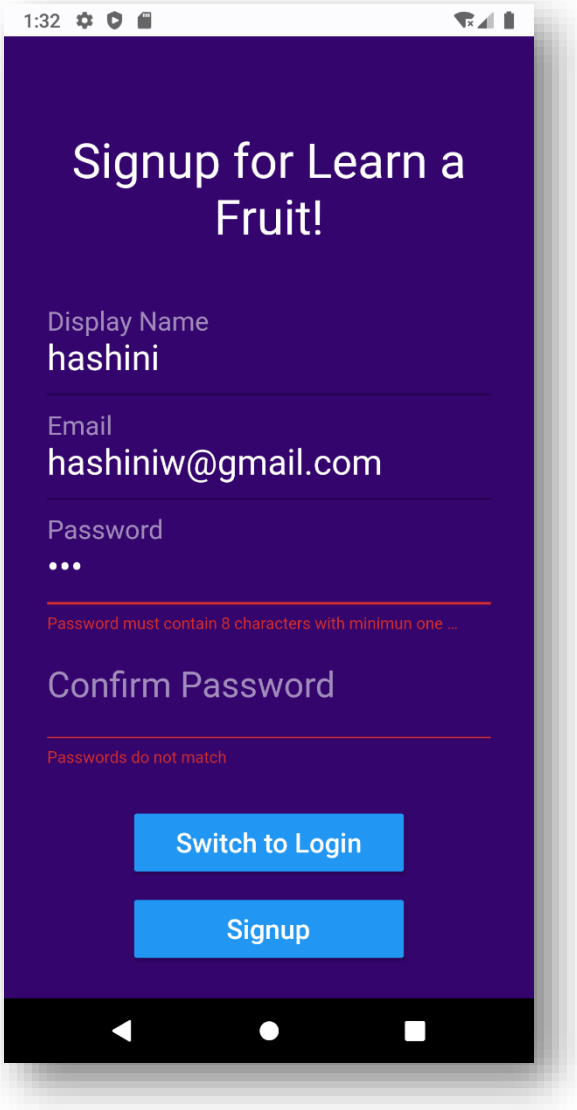

Figure 2.4: Validation called when invalid password is entered

Password must be of 8 characters minimum including minimum one uppercase letter, minimum one lowercase letter and minimum one numeric value.

Also, both the password and confirm password fields validated in order to signup successfully.

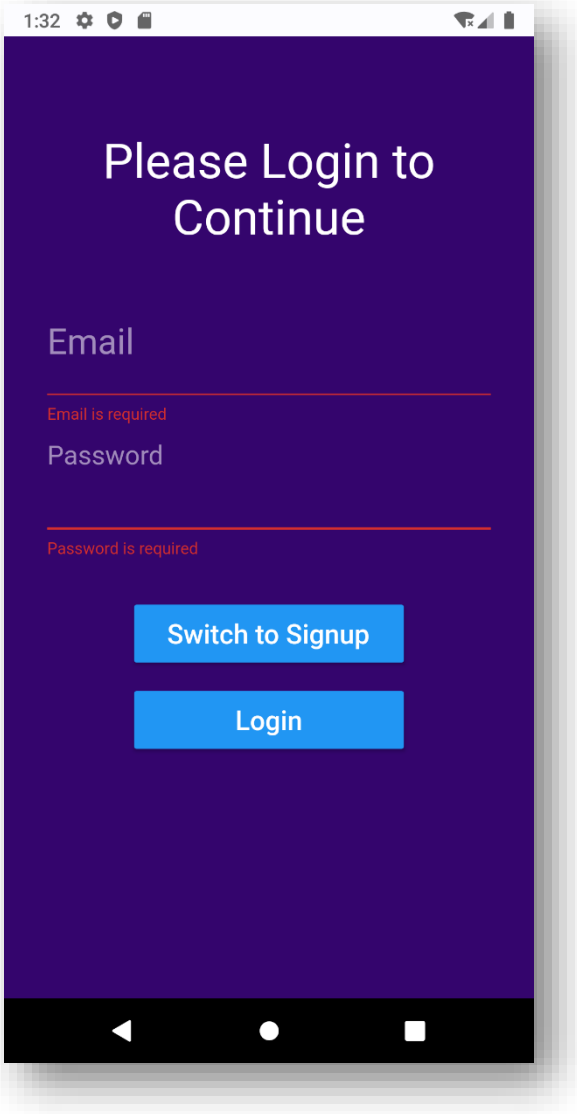

Figure 2.5: User cannot login without having a valid authentication

Valid user authentication has been added; hence the user cannot just directly log into the app without user authentication.

2.3 Home Screen

In home screen a carousel slider has been added, in order to give user, the clear overview of what the app does. Plus, this is adding a great attribute to the UI which ultimately gives user a greater user experience. These simpler UI layout leads the application to be more user-friendly.

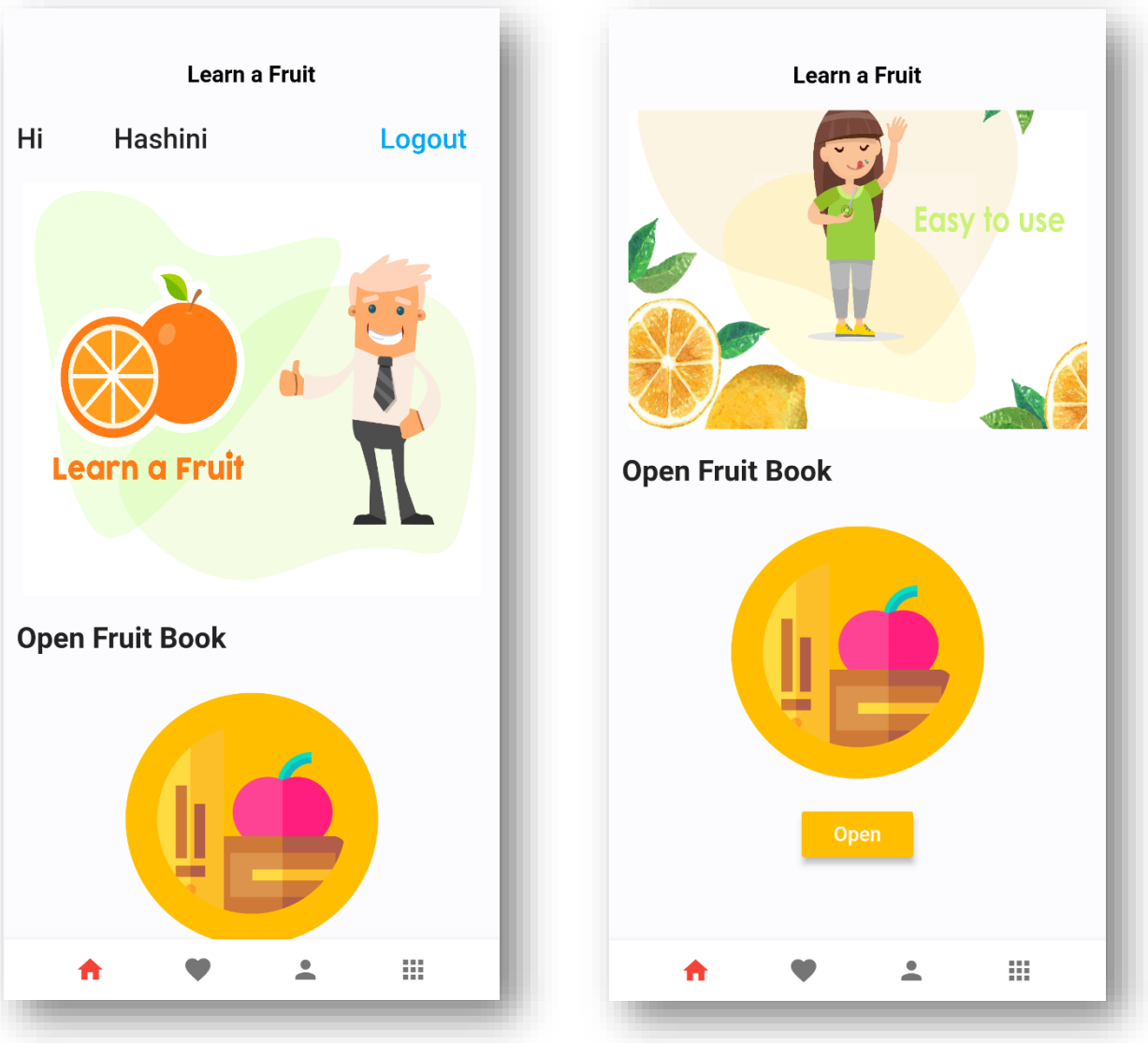

Figure 3: Home screen of Learn a Fruit

2.4 Favorite Fruit Collection Screen

We can add a favorite fruit collection by clicking the heart shaped icon (favorite icon) in this app. This is the general bookmark adding functionality. From the added fruit collection, we can select some fruits and add them to the favorite fruit collection. As well we can remove them when we don't want it to be in the favorite fruit collection.

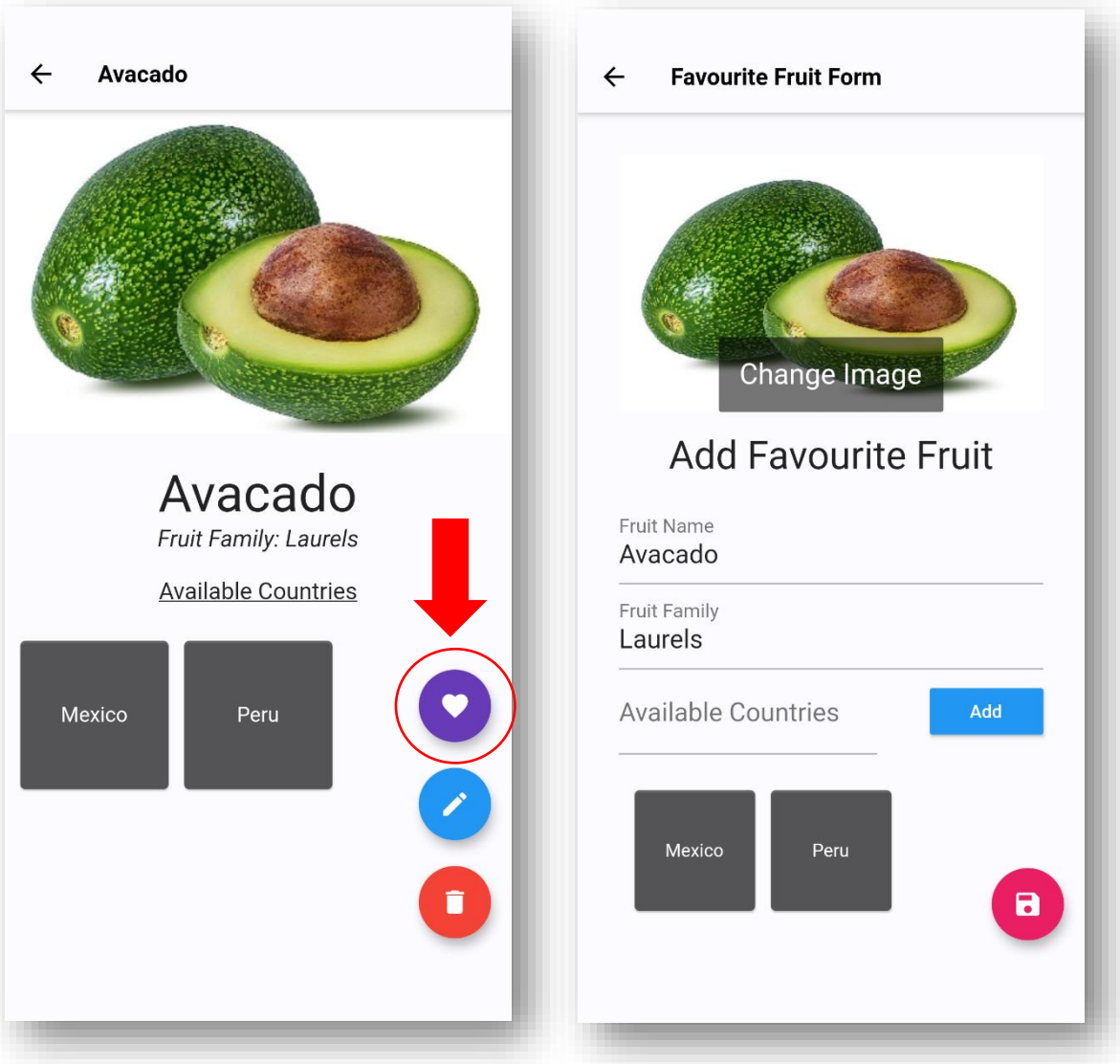

Figure 4: Adding fruits to favorite fruit collection

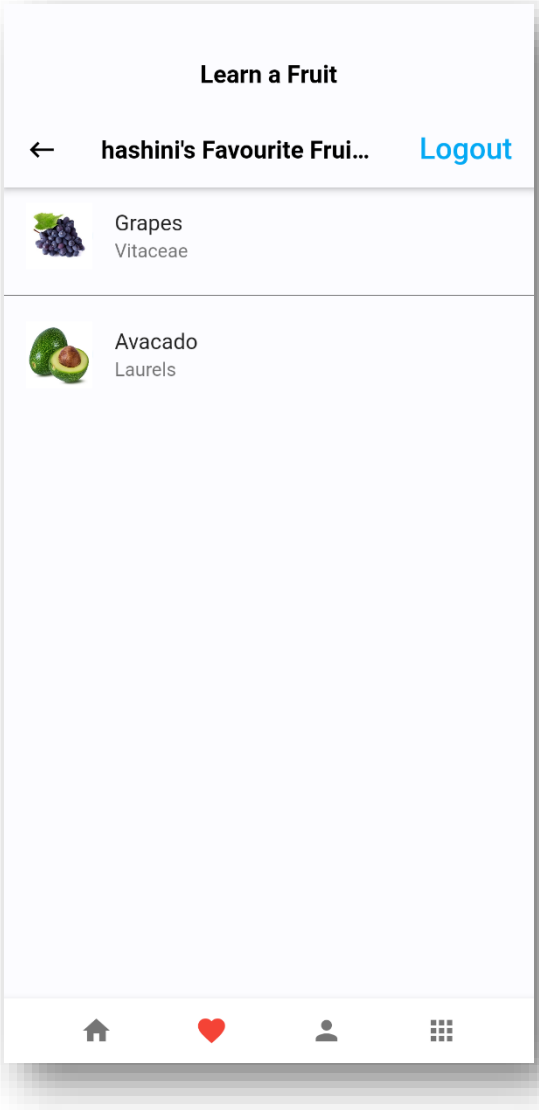

Figure 4.1: UI screen of the favorite fruit collection after adding favorite fruits

If we go to the database and look at the favorite fruit collection, (FFruit), we can observe that it has been created successfully and our favorite fruits have been added to the database.

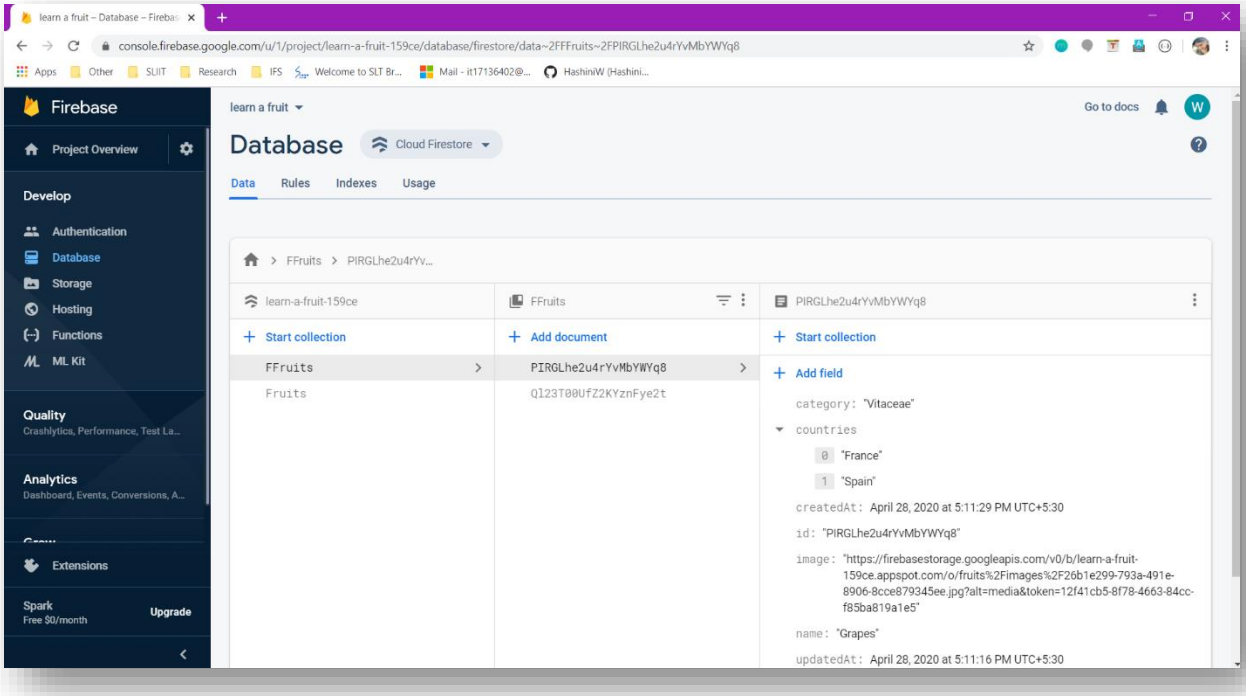

Figure 4.2: Created FFruit collection (favorite fruit collection) in firebase

As well adding the fruit, we can remove the added favorite fruit from the favorite fruit collection. We can remove them when we don't want it to be in the favorite fruit collection.

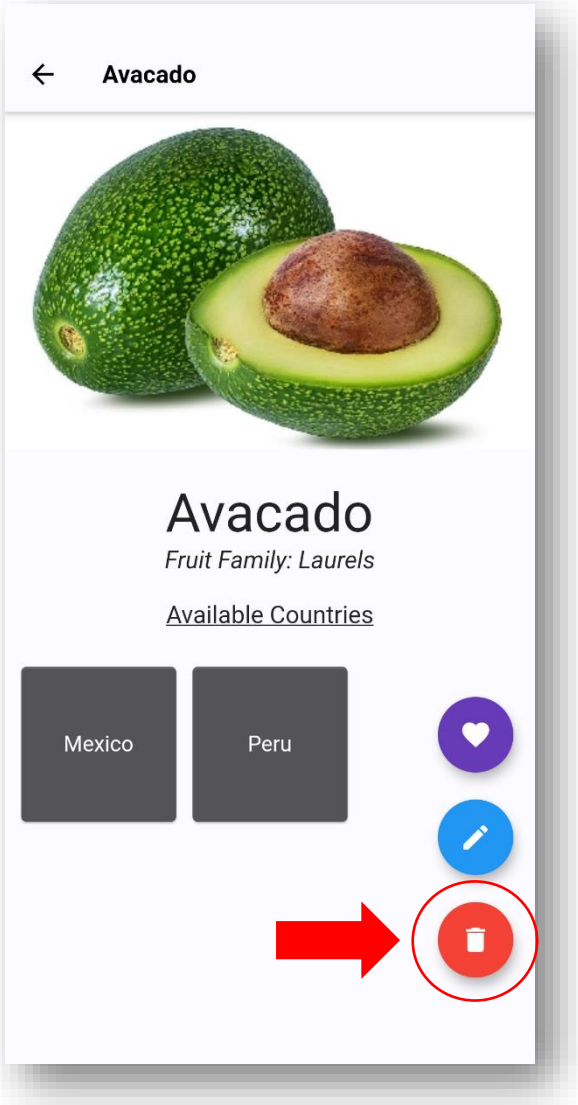

Figure 5: Removing fruits to favorite fruit collection

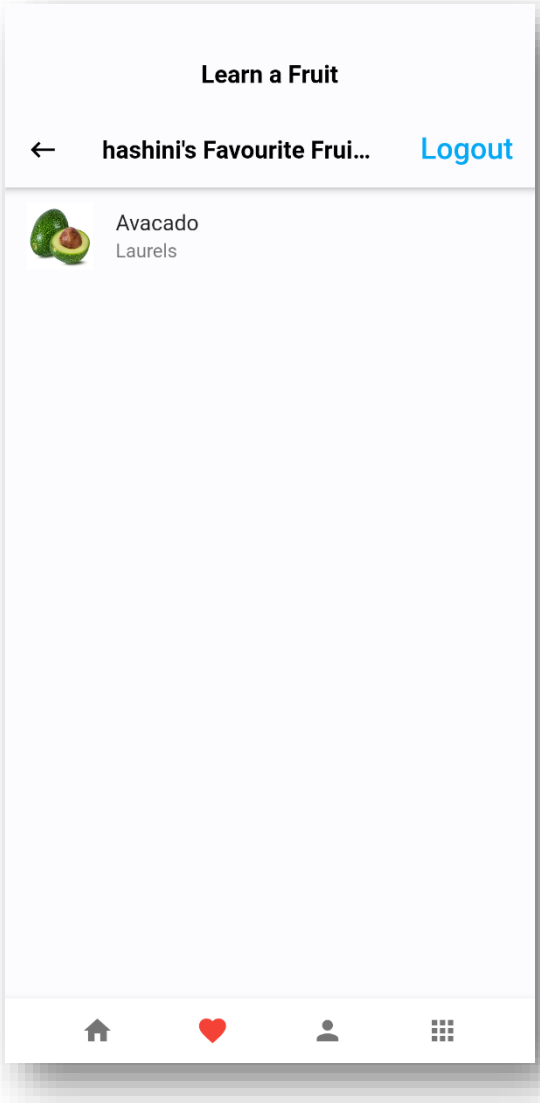

Figure 5.1: UI screen of the favorite fruit collection after removing a favorite fruit

If we go to the database and look at the favorite fruit collection, (FFruit), we can observe that the fruit we have removed has been removed from the firebase collection also.

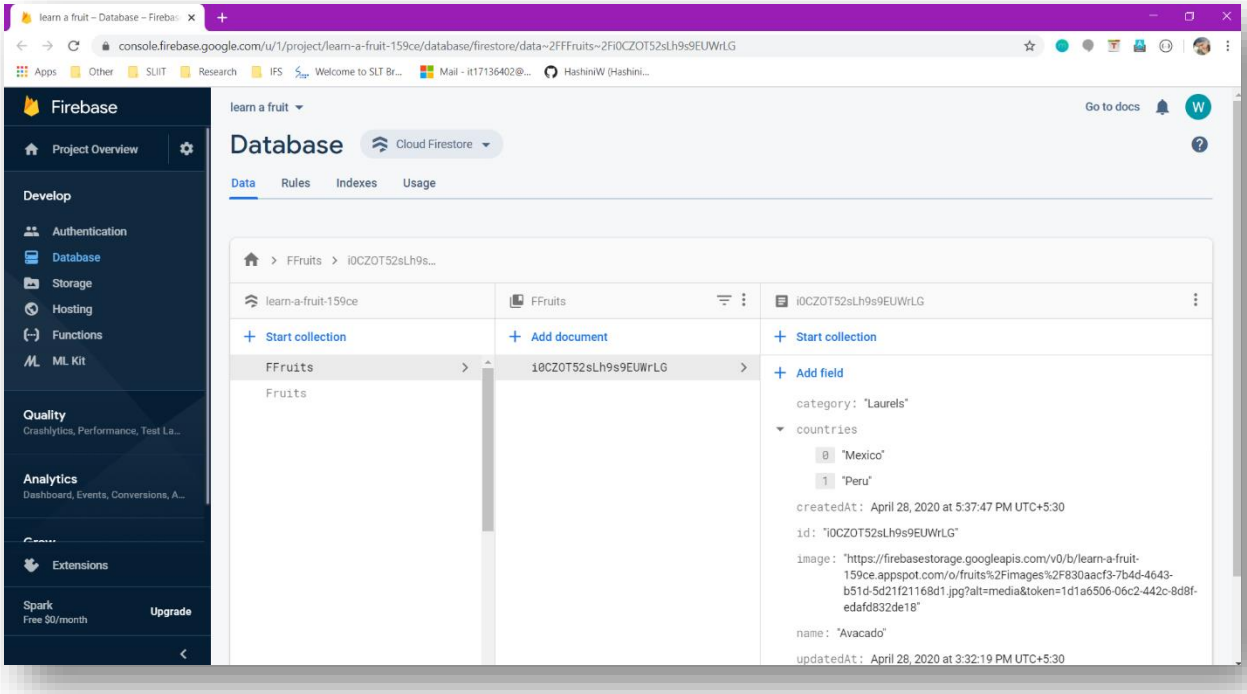

Figure 5.2: FFruit collection (favorite fruit collection) in firebase after removing a favorite fruit

2.5 Profile Details Screen

User can alter their profile by adding a profile picture for their user profiles. Or else they can change their display name as they wish.

To select the user profile picture, camera access has been granted. As well the local file storage access also has been granted. User can either directly take a photo by opening their camera or else they can access their gallery and add a profile picture.

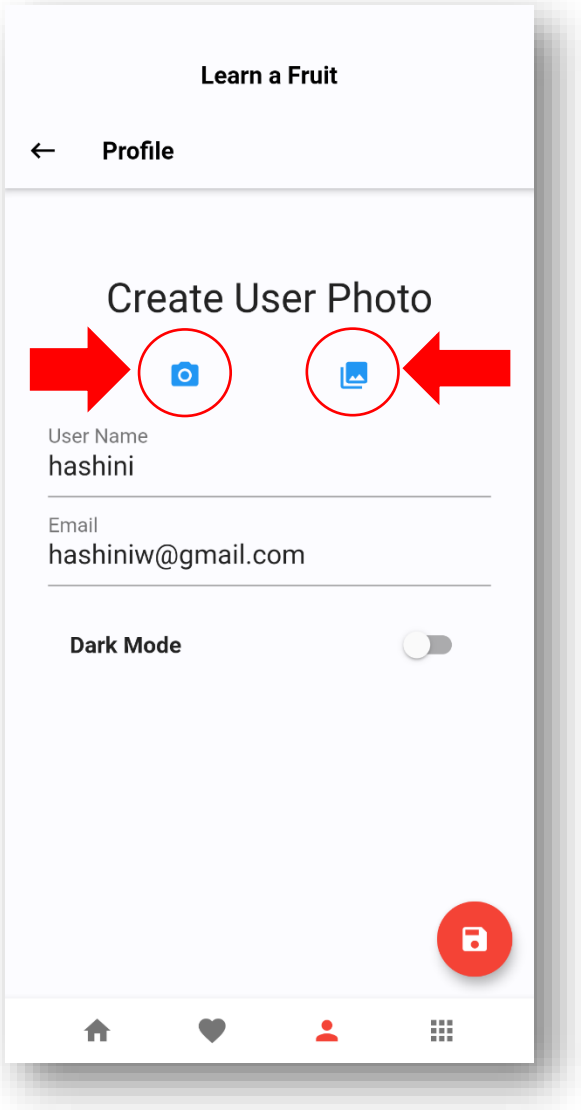

Figure 6: UI screen of Profile page

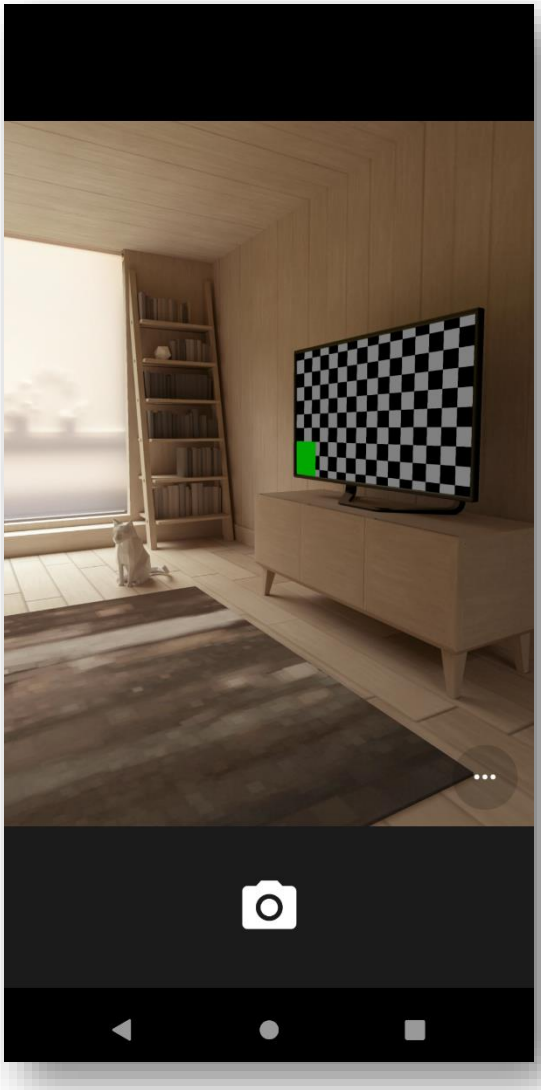

Figure 6.1: Accessing the camera in order to take a profile picture

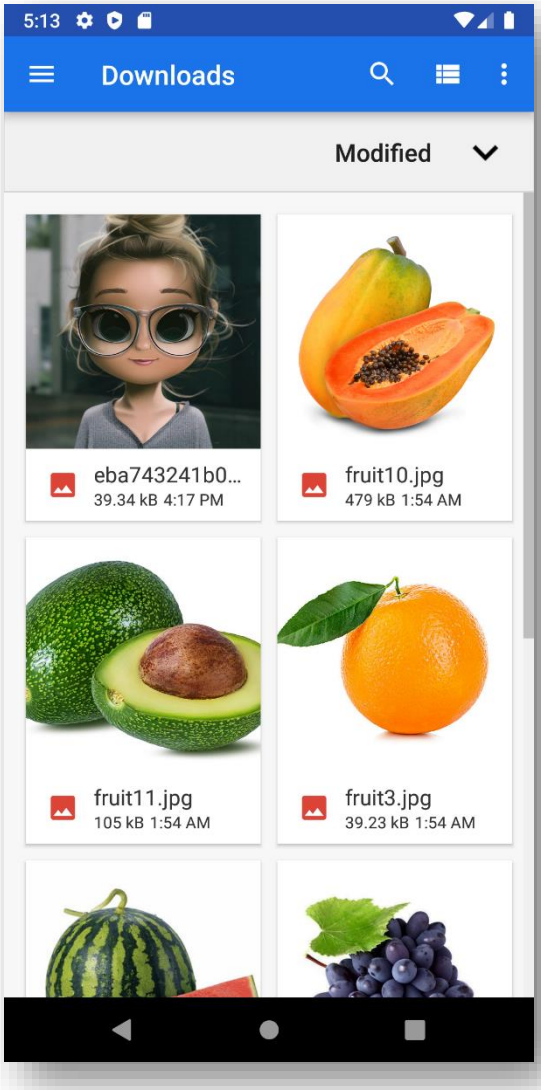

Figure 6.2: Accessing the local file storage in order to select a profile picture

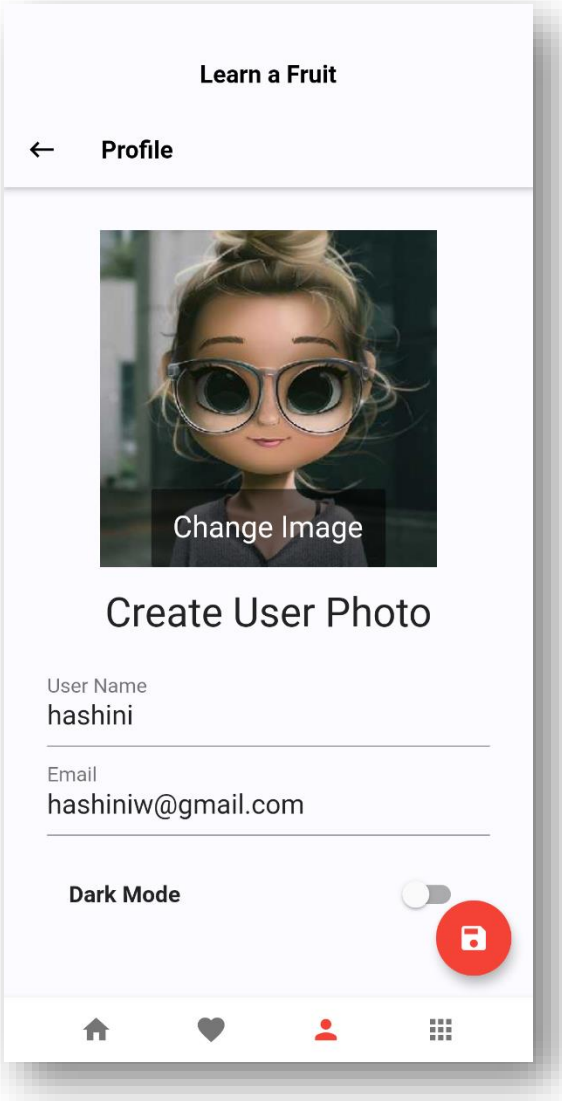

Figure 6.3: Saving the profile picture for the current user

After adding an image, if you wish to change it you can simply press ‘Change Image’ button and choose an image as you wish.

If we go to the database, we can observe that the Users collection has been created and our profile picture has been successfully added.

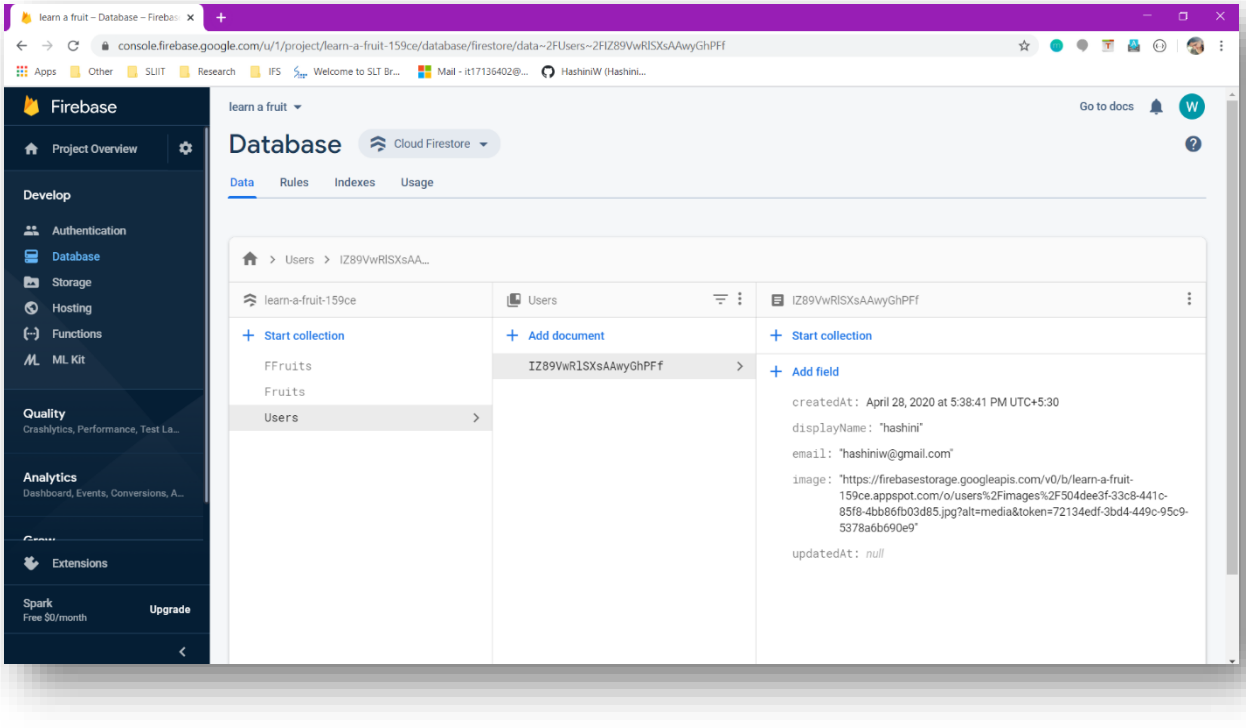

Figure 6.4: Users collection after adding a profile picture in Firebase

3. References

[1]"Flutter-Image Upload.", github.com. [Online]. Available: https://github.com/whatsupcoders/Flutter-ImageUpload. [Accessed: 02 – Apr - 2020].

[2] Julian Curry “Code with Curry”, https://github.com/JulianCurrie/CwC_Flutter [Accessed:20-Jan-2020]

[3] Julian Curry “Code with Curry”, https://www.youtube.com/watch?v=bjMw89L61FI [Accessed: 14-Mar-2020]

[4] TechieBlossom “SideBarNavigationalPage” https://github.com/TechieBlossom/sidebar_animation_flutter [Accessed: 11-Mar-2020]

[5] Restaurant UI Kit “Flutter Front UI Design” https://apkpure.com/flutter-mobile-restaurant-ui-kit/com.jideguru.restaurant_ui_kit [Accessed: 12-Mar-2020]

4. Appendix

4.1 Code Implementation

LoginPageDisplay.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference 1 : //https://stackoverflow.com/questions/56253787/how-to-handle-
textfield-validation-in-password-in-flutter
Reference 2 : https://github.com/JulianCurrie/CwC_Flutter
*/

import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import '../Crudmodel/UserCrudModel.dart';
import '../CrudControllers/authentication_Controller.dart';

enum AuthMode { Signup, Login }

class LoginPageDisplayUI extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _LoginPageDisplayUIState();
  }
}

class _LoginPageDisplayUIState extends State<LoginPageDisplayUI> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _passwordController = new TextEditingController();
  AuthMode _authMode = AuthMode.Login;

  UserCrudModel _user = UserCrudModel();

  @override
  void initState() {
    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context, listen: false);
    initializeCurrentUser(authNotifier);
    super.initState();
  }

  //submitting the user authentication forms
  void _submitForm() {
    if (!_formKey.currentState.validate()) {
      return;
```

```
    }

    _formKey.currentState.save();

    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context, listen: false);

    if (_authMode == AuthMode.Login) {
      login(_user, authNotifier);
    } else {
      signup(_user, authNotifier);
    }
  }

  //validating the display name filed
  Widget _buildDisplayNameField() {
    return TextFormField(
      decoration: InputDecoration(
        labelText: "Display Name",
        labelStyle: TextStyle(color: Colors.white54),
      ),
      keyboardType: TextInputType.text,
      style: TextStyle(fontSize: 26, color: Colors.white),
      cursorColor: Colors.white,
      validator: (String value) {
        if (value.isEmpty) {
          return 'Display Name is required';
        }

        return null;
      },
      onSaved: (String value) {
        _user.displayName = value;
      },
    );
  }

  //validating the email field
  Widget _buildEmailField() {
    return TextFormField(
      decoration: InputDecoration(
        labelText: "Email",
        labelStyle: TextStyle(color: Colors.white54),
      ),
      keyboardType: TextInputType.emailAddress,
      style: TextStyle(fontSize: 26, color: Colors.white),
      cursorColor: Colors.white,
      validator: (String value) {
```

```
        if (value.isEmpty) {
          return 'Email is required';
        }
        //using a RegExp to recognize an email
        if (!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 'Please enter a valid email address';
        }

        return null;
      },
      onSaved: (String value) {
        _user.email = value;
      },
    );
  }
  //validating the password filed
  Widget _buildPasswordField() {
    return TextFormField(
      decoration: InputDecoration(
        labelText: "Password",
        labelStyle: TextStyle(color: Colors.white54),
      ),
      style: TextStyle(fontSize: 26, color: Colors.white),
      cursorColor: Colors.white,
      obscureText: true,
      controller: _passwordController,
      validator: (String value) {
        if (value.isEmpty) {
          return 'Password is required';
        }
        //https://stackoverflow.com/questions/56253787/how-to-handle-textfield-
validation-in-password-in-flutter
        /*I have created my own RegExp by analysing the above example. Here I'm
checking a 8 character password with minimum one uppercase character,
        minimum one lowercase character and minimum one numeric value*/
        if (!RegExp(
                r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]).{8,}$')
            .hasMatch(value)) {
          return 'Password must contain 8 characters with minimun one uppsercase
 letter, minimum one lowercase letter, minimum one numeric value';
        }

        return null;
      },
      onSaved: (String value) {
```

```
      _user.password = value;
    },
  );
}

//validating the password confirmation
Widget _buildConfirmPasswordField() {
  return TextFormField(
    decoration: InputDecoration(
      labelText: "Confirm Password",
      labelStyle: TextStyle(color: Colors.white54),
    ),
    style: TextStyle(fontSize: 26, color: Colors.white),
    cursorColor: Colors.white,
    obscureText: true,
    validator: (String value) {
      if (_passwordController.text != value) {
        return 'Passwords do not match';
      }

      return null;
    },
  );
}

//UI layout
@override
Widget build(BuildContext context) {
  print("Building login screen");

  return Scaffold(
    body: Container(
      constraints: BoxConstraints.expand(
        height: MediaQuery.of(context).size.height,
      ),
      decoration: BoxDecoration(color: Color(0xff34056D)),
      child: Form(
        autovalidate: true,
        key: _formKey,
        child: SingleChildScrollView(
          child: Padding(
            padding: EdgeInsets.fromLTRB(32, 96, 32, 0),
            child: Column(
              children: <Widget>[
                Text(

                  //checking the authmode to display the valid screen
```

```
                    _authMode == AuthMode.Login ? 'Please Login to Continue' : '
Signup for Learn a Fruit!',
                    textAlign: TextAlign.center,
                    style: TextStyle(fontSize: 36, color: Colors.white),
                  ),
                  SizedBox(height: 32),
                  _authMode == AuthMode.Signup ? _buildDisplayNameField() : Cont
ainer(),
                  _buildEmailField(),
                  _buildPasswordField(),
                  _authMode == AuthMode.Signup ? _buildConfirmPasswordField() :
Container(),
                  SizedBox(height: 32),
                  ButtonTheme(
                    minWidth: 200,
                    child: RaisedButton(
                      padding: EdgeInsets.all(10.0),
                      child: Text(
                        'Switch to ${_authMode == AuthMode.Login ? 'Signup' : 'L
ogin'}',
                        style: TextStyle(fontSize: 20, color: Colors.white),
                      ),
                      onPressed: () {
                        setState(() {
                          _authMode =
                              _authMode == AuthMode.Login ? AuthMode.Signup : Au
thMode.Login;
                        });
                      },
                    ),
                  ),
                  SizedBox(height: 16),
                  ButtonTheme(
                    minWidth: 200,
                    child: RaisedButton(
                      padding: EdgeInsets.all(10.0),
                      onPressed: () => _submitForm(),
                      child: Text(
                        _authMode == AuthMode.Login ? 'Login' : 'Signup',
                        style: TextStyle(fontSize: 20, color: Colors.white),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
```

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

MainPageDisplay.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference-1 : https://apkpure.com/flutter-mobile-restaurant-
ui-kit/com.jideguru.restaurant_ui_kit
Reference-2 : https://github.com/whatsupcoders/Flutter-ImageUpload
*/

import 'dart:io';

import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import 'package:finalproject/CrudControllers/authentication_Controller.dart';
import 'package:flutter/material.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'package:provider/provider.dart';
import 'FruitBook.dart';
import 'LoginPageDisplay.dart';

class HomeScreenUI extends StatefulWidget {
  @override
  _HomeScreenUIState createState() => _HomeScreenUIState();
}

class _HomeScreenUIState extends State<HomeScreenUI> with AutomaticKeepAliveClie
ntMixin<HomeScreenUI>{
  List<T> map<T>(List list, Function handler) {
    List<T> result = [];
    for (var i = 0; i < list.length; i++) {
      result.add(handler(i, list[i]));
    }

    return result;
  }

  int _current = 0;

  //listing the slider images
  List imageList = [
    'assets/intro1.png',
    'assets/intro2.png',
    'assets/intro3.png'
```

```
  ];

  File _image;

  @override
  Widget build(BuildContext context) {
    super.build(context);

    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context);

    return Scaffold(
      body: Padding(
        padding: EdgeInsets.fromLTRB(10.0,0,10.0,0),
        child: ListView(
          children: <Widget>[
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: <Widget>[
                Text('Hi',
                  style: TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.w600,
                  ),
                ),
              //  SizedBox(width: 10.0),
                Text(

                  //showing user name
                  authNotifier.user != null ?  authNotifier.user.displayName : "
Fruity!",
                  style: TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                SizedBox(width: 10.0),

                FlatButton(
                  onPressed: (){
                    Navigator.of(context).push(
                      MaterialPageRoute(
                        builder: (BuildContext context){
                          signout(authNotifier);
                          return LoginPageDisplayUI();
                        },
                      ),
                    );
```

```
            },
            child: Text(

              //logginout from the app
              "Logout",
              style: TextStyle(fontSize: 22, color: Colors.lightBlue),
            ),
          ),
        ],
      ),
      SizedBox(height: 10.0),

      //Slider Here
      CarouselSlider(
        height: MediaQuery.of(context).size.height/2.4,
        items: imageList.map((i){
          return Builder(
            builder: (BuildContext context) {
              return Container(
                width: MediaQuery.of(context).size.width,
                margin: EdgeInsets.symmetric(horizontal: 5.0),
                decoration: BoxDecoration(
                  color: Colors.white
                ),
                child: Image.asset(i, fit: BoxFit.fill,),
              );
            },
          );
        }).toList(),
        autoPlay: true,
        viewportFraction: 1.0,
        onPageChanged: (index) {
          setState(() {
            _current = index;
          });
        },
      ),
      SizedBox(height: 20.0),
      Text(
        "Open Fruit Book",
        style: TextStyle(
          fontSize: 23,
          fontWeight: FontWeight.w800,
        ),
      ),
      SizedBox(height: 10.0),

      Container(
```

```
              child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: <Widget>[
                  SizedBox(height: 20.0,),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                      Align(
                        alignment: Alignment.center,
                        child: CircleAvatar(
                          radius: 100,
                          backgroundColor: Color(0xffffbf00),
                          child: ClipOval(
                            child:SizedBox(
                              width: 150.0,
                              height: 150.0,
                              child: (_image != null)?Image.file(_image, fit: Bo
xFit.fill,)
                              :Image.asset(
                                'assets/book.png',
                                width: 600.0,
                                height: 240.0,
                                fit: BoxFit.cover,
                              ),
                            )
                          ),
                        ),
                      ),
                    ],
                  ),
                  SizedBox(
                    height: 20.0,
                  ),
                ],
              ),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                RaisedButton(
                  color: Color(0xffffbf00),
                  onPressed: (){
                    Navigator.of(context).push(
                      MaterialPageRoute(
                        builder: (BuildContext context){
                          return FruitBook();
                        },
                      ),
```

```
                  );
                },
                elevation: 4.0,
                splashColor: Colors.blueGrey,
                child: Text(
                  'Open',
                  style: TextStyle(color: Colors.white, fontSize: 16.0),
                ),
              ),
            ],
          ),
          SizedBox(height: 20.0),
          SizedBox(height: 10.0),
          SizedBox(height: 30),

        ],
      ),
    ),
  );
 }

 @override
 bool get wantKeepAlive => true;
}
```

ProfilePageDisplay.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference-1 : https://github.com/JulianCurrie/CwC_Flutter
Reference-2 : https://www.youtube.com/watch?v=bjMw89L61FI
Reference-3 : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4 : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
*/

import 'dart:io';

import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import 'package:finalproject/Crudmodel/UserCrudModel.dart';
import 'package:finalproject/CrudControllers/authentication_Controller.dart';
import 'package:finalproject/CrudControllers/Fruit_Controller.dart';
import 'package:finalproject/LearnAFruitproviders/LearnAFruit_provider.dart';
import 'package:finalproject/LearnAFruitUtilities/constColourAttributer.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
```

```dart
import 'package:provider/provider.dart';
import 'UICollectionHandler.dart';

//creating the class to input user details for adding the details of the user
class ProfileUI extends StatefulWidget {

  //checking whether needs to update or not
  final bool isUpdating;

  //loading the details to the constructor
  ProfileUI({@required this.isUpdating});

  //handle the user form state
  @override
  _ProfileUIState createState() => _ProfileUIState();
}

//creating the class to input user details for adding the details of the user
class _ProfileUIState extends State<ProfileUI> {

  //declaring the global form key to maintain form state
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  //declaring the global scaffold key to maintain scaffold state
  final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();

  //creating the current user details object
  UserCrudModel _currentUser;
  String displayName;

  //creating the image url for the user
  String _imageUrl;

  //creating the image file to store in cloud store bucket
  File _imageFile;

  //maintaining the state of the user controller
  @override
  void initState() {
    super.initState();
    FruitController fruitNotifier = Provider.of<FruitController>(context, listen
: false);

    //checking if current user object is null then load the current details
    if (fruitNotifier.currentUser != null) {
      _currentUser = fruitNotifier.currentUser;
    } else {
      _currentUser = UserCrudModel();
```

```
    }
    _imageUrl = _currentUser.image;
  }

  //check if the the file and image url is null then show in the placeholder as
image placeholder otherwise print in console as showing image from local file
  _showImage() {
    if (_imageFile == null && _imageUrl == null) {
      return Text(" ");
    } else if (_imageFile != null) {
      print('showing image from local file');

      //designing the ui level
      return Stack(
        alignment: AlignmentDirectional.bottomCenter,
        children: <Widget>[
          Image.file(
            _imageFile,
            fit: BoxFit.cover,
            height: 250,
          ),
          FlatButton(
            padding: EdgeInsets.all(16),
            color: Colors.black54,
            child: Text(
              'Change Image',
              style: TextStyle(color: Colors.white, fontSize: 22, fontWeight: Fo
ntWeight.w400),
            ),
            onPressed: () => _getLocalImage(),
          )
        ],
      );
    } else if (_imageUrl != null) {
      print('showing image from url');
      return Stack(
        alignment: AlignmentDirectional.bottomCenter,
        children: <Widget>[
          Image.network(
            "https://images.unsplash.com/photo-1502164980785-
f8aa41d53611?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=500&q=60",
            width: MediaQuery.of(context).size.width,
            fit: BoxFit.cover,
            height: 250,
          ),
          FlatButton(
            padding: EdgeInsets.all(16),
```

```
            color: Colors.black54,
            child: Text(
              'Change Image',
              style: TextStyle(color: Colors.white, fontSize: 22, fontWeight: Fo
ntWeight.w400),
            ),
            onPressed: () => _getLocalImage(),
          )
        ],
      );
    }
  }

  //getting the image file from image gallery to load image in the placeholder
  _getLocalImage() async {
    File imageFile =
    await ImagePicker.pickImage(source: ImageSource.gallery, imageQuality: 50, m
axWidth: 400);

    if (imageFile != null) {
      setState(() {
        _imageFile = imageFile;
      });
    }
  }

  //getting the image file from mobile camera to load image in the placeholder
  _openCamera() async {
    File imageFile =
    await ImagePicker.pickImage(source: ImageSource.camera, imageQuality: 100, m
axWidth: 400);

    if (imageFile != null) {
      setState(() {
        _imageFile = imageFile;
      });
    }
  }

  //handle the name field with validation using text form controller
  Widget _buildNameField() {
    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context);
    return TextFormField(
      decoration: InputDecoration(labelText: 'User Name'),
      initialValue:  authNotifier.user != null ?  authNotifier.user.displayName
: "Feed",
      keyboardType: TextInputType.text,
```

```
      style: TextStyle(fontSize: 20),
      validator: (String value) {
        if (value.isEmpty) {
          return 'Name is required';
        }
        return null;
      },
      onSaved: (String value) {
        _currentUser.displayName = value;
      },
    );
  }

  //handle the email field with validation using text form controller
  Widget _buildEmailField() {
    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context);
    return TextFormField(
      decoration: InputDecoration(labelText: 'Email'),
      initialValue: authNotifier.user != null ?  authNotifier.user.email : "Null
",
      keyboardType: TextInputType.text,
      style: TextStyle(fontSize: 20),
      validator: (String value) {
        if (value.isEmpty) {
          return 'Email is required';
        }

        if (!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 'Please enter a valid email address';
        }

        return null;
      },
      onSaved: (String value) {
        _currentUser.email = value;
      },
    );
  }

  //sending the uploaded user details to add to the cloud store
  _onUserUploaded(UserCrudModel user) {
    FruitController fruitNotifier = Provider.of<FruitController>(context, listen
: false);
    fruitNotifier.addUser(user);
```

```
    Navigator.pop(context);
  }

   //save the current state of the form text forms
  _saveUser() {
    print('saveUser Called');
    if (!_formKey.currentState.validate()) {
      return;
    }

    _formKey.currentState.save();

    print('form saved');

    uploadUserAndImage(_currentUser, widget.isUpdating, _imageFile, _onUserUploa
ded);

    print("displayName: ${_currentUser.displayName}");
    print("email: ${_currentUser.email}");
    print("_imageFile ${_imageFile.toString()}");
    print("_imageUrl $_imageUrl");
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      appBar: AppBar(
          leading: IconButton(
            icon: Icon(
              Icons.keyboard_backspace,
            ),
            onPressed: ()=>  Navigator.of(context).push(MaterialPageRoute(builde
r: (BuildContext context) {
              return UICollectionHandler();
            })),
          ),
          title: Text('Profile')),
      body: SingleChildScrollView(
        padding: EdgeInsets.all(32),
        child: Form(
          key: _formKey,
          autovalidate: true,
          child: Column(children: <Widget>[
            _showImage(),
            SizedBox(height: 16),
            Text(
              widget.isUpdating ? "Edit User" : "Create User Photo",
```

```
            textAlign: TextAlign.center,
            style: TextStyle(fontSize: 30),
          ),
          SizedBox(height: 16),
          _imageFile == null && _imageUrl == null
              ? Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: <Widget>[
              Align(
                alignment: Alignment.centerLeft,
                child: Container(
                  child: Column(children: <Widget>[
                    Align(alignment: Alignment.centerLeft,
                      child: IconButton(
                        icon: Icon(
                          Icons.photo_camera,
                          color: Colors.blue,
                        ),
                        onPressed: () {
                          _openCamera();
                        },),
                    ),
                  ],),
                ),
              ),
              Align(
                alignment: Alignment.centerRight,
                child: Container(
                  child: IconButton(
                    icon: Icon(
                      Icons.photo_library,
                      color: Colors.blue,
                    ),
                    onPressed: () {
                      _getLocalImage();
                    },),
                ),
              )
            ],
          )
              : SizedBox(height: 0),
          _buildNameField(),
          _buildEmailField(),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: <Widget>[

            ],
```

```
            ),
            SizedBox(height: 16),

            //reference4: https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
            //this method switches the theme type to dark mode to white mode vis
e versa using app
            ListTile(
              title: Text(
                "Dark Mode",
                style: TextStyle(
                  fontSize: 17,
                  fontWeight: FontWeight.w700,
                ),
              ),

              trailing: Switch(
                value: Provider.of<LearnAFruitProvider>(context).theme == Consta
nts.lightTheme
                    ? false
                    : true,
                onChanged: (v) async{
                  if (v) {
                    Provider.of<LearnAFruitProvider>(context, listen: false)
                        .setTheme(Constants.darkTheme, "dark");
                  } else {
                    Provider.of<LearnAFruitProvider>(context, listen: false)
                        .setTheme(Constants.lightTheme, "light");
                  }
                },
                activeColor: Theme.of(context).accentColor,
              ),
            ),
          ]),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          FocusScope.of(context).requestFocus(new FocusNode());
          _saveUser();
        },
        child: Icon(Icons.save),
        foregroundColor: Colors.white,
      ),
    );

  }
}
```

Fruit_Controller.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya, W.G.M.V.S Wijesundara -
 IT17035118
Description : Creating the firebase fruit object to set all the data to the noti
fy classes
Reference-1 :  https://github.com/JulianCurrie/CwC_Flutter
Reference-2 : https://www.youtube.com/watch?v=bjMw89L61FI
Reference-3 : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4 : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
*/


import 'dart:collection';
import 'package:finalproject/Crudmodel/FruitCrudModel.dart';
import 'package:finalproject/Crudmodel/UserCrudModel.dart';
import 'package:flutter/cupertino.dart';

//creating the Fruit details list  controller class for fruit object handling
class FruitController with ChangeNotifier {
  //creating a list object to store private fruit details
  List<FruitCrudModel> _fruitList = [];
  //creating a list object to store private user details
  List<UserCrudModel> _userList= [];
  //creating a object to store private current loading fruit details
  FruitCrudModel _currentFruit;
  //creating a  object to store private after login user details in current sess
ion
  UserCrudModel _currentUser;
  //creating a list view to get fruit details
  UnmodifiableListView<FruitCrudModel> get fruitList => UnmodifiableListView(_fr
uitList);
  //creating a list view to get user details
  UnmodifiableListView<UserCrudModel> get userList => UnmodifiableListView(_user
List);
  //getting the current fruit details
  FruitCrudModel get currentFruit => _currentFruit;
  //getting the current user details
  UserCrudModel get currentUser => _currentUser;
```

```
  //creating the fruit details setter object to set the fruit data
  set fruitList(List<FruitCrudModel> fruitList) {
    _fruitList = fruitList;
    notifyListeners();
  }
  //creating the user details setter object to set the user data
  set userList(List<UserCrudModel> userList) {
    _userList = userList;
    notifyListeners();
  }
  //creating the current loading fruit details setter object to set its relevant
 fruit data
  set currentFruit(FruitCrudModel fruit) {
    _currentFruit = fruit;
    notifyListeners();
  }

 //creating the current loading user details setter object to set its relevant u
ser data after authentication
  set currentUser(UserCrudModel user) {
    _currentUser = user;
    notifyListeners();
  }
  //creating the adding fruit method to add fruit details to store
  addFruit(FruitCrudModel fruit) {
    _fruitList.insert(0, fruit);
    notifyListeners();
  }
  //creating the adding user method to add user details to store
  addUser(UserCrudModel user) {
    _userList.insert(0, user);
    notifyListeners();
  }
  //creating the deleting fruit method to delete fruit details from fire cloud s
tore
  deleteFruit(FruitCrudModel fruit) {
    _fruitList.removeWhere((_fruit) => _fruit.id == fruit.id);
    notifyListeners();
  }
  //creating the deleting fruit method to delete favorite fruit details from fir
e cloud store
  deleteFavouriteFruit(FruitCrudModel fruit) {
    _fruitList.removeWhere((_fruit) => _fruit.id == fruit.id);
    notifyListeners();
  }
}
```

UserCrudModel.dart

```dart
/*
Author      : IT17136402 -
 W.M.H.B. Warnakulasooriya, W.G.M.V.S Wijesundara  IT17035118
Description : Creating the fruit crud model to perform crud operations
Reference-1 : https://github.com/JulianCurrie/CwC_Flutter
Reference-2 : https://www.youtube.com/watch?v=bjMw89L61FI
Reference-3 : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4 : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
 */

import 'package:cloud_firestore/cloud_firestore.dart';

//creating the class to create the attributes for the user crud model to handle
the user details
class UserCrudModel {
  //declare string name for the each user crud  to display after login in main s
creen
  String displayName;
  //declare unique string email for the each user crud
  String email;
  //declare string password for the each user crud
  String password;
  //declare string image for the each user crud
  String image;
  //declare creating time to record the created time for the each user crud
  Timestamp createdAt;
  //declare updating time to record the updated time for the each user crud
  Timestamp updatedAt;

  //default constructor implementation
  UserCrudModel();


  //mapping the string based json data to document using the user crud model
  UserCrudModel.fromMap(Map<String, dynamic> data) {
    displayName = data['displayName'];
    email = data['email'];
    image = data['image'];
    createdAt = data['createdAt'];
    updatedAt = data['updatedAt'];
  }
  //returning the created data using user crud model
  Map<String, dynamic> toMap() {
    return {
      'displayName': displayName,
      'email': email,
```

```
      'image': image,
      'createdAt': createdAt,
      'updatedAt': updatedAt
    };
  }


}
```

Favorite_Screen.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference-1 : https://github.com/whatsupcoders/Flutter-ImageUpload
Reference-2 : https://apkpure.com/flutter-mobile-restaurant-
ui-kit/com.jideguru.restaurant_ui_kit
*/

import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import 'package:finalproject/CrudControllers/authentication_Controller.dart';
import 'package:finalproject/CrudControllers/Fruit_Controller.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'FavouriteDetails.dart';
import 'details.dart';
import 'LoginPageDisplay.dart';
import 'UICollectionHandler.dart';

//creating the class to surf as the main screen for to go the favorite fruit lis
t  and manage the state of the page
class FavoriteScreen extends StatefulWidget {
  @override
  _FavouriteFruitBookState createState() => _FavouriteFruitBookState();
}

//creating the class to surf as the main screen for to go the favorite fruit lis
t
class _FavouriteFruitBookState extends State<FavoriteScreen> {

  //maintaining the state of the fruit controller to get favorite fruits list
  @override
  void initState() {
    FruitController fruitNotifier = Provider.of<FruitController>(context, listen
: false);
    getFavouriteFruits(fruitNotifier);
```

```
    super.initState();
  }

//calling the authentication class to get the display name
//calling the favorite fruit controller class to get the list of favorite fruits
 to display
  @override
  Widget build(BuildContext context) {
    AuthenticationController authNotifier = Provider.of<AuthenticationController
>(context);
    FruitController fruitNotifier = Provider.of<FruitController>(context);

    //showing the list of favorite fruits in list  view builder
    Future<void> _refreshList() async {
      getFavouriteFruits(fruitNotifier);
    }

    print("Opening Favourite Fruit Book");
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: Icon(
            Icons.keyboard_backspace,
          ),
          onPressed: ()=>  Navigator.of(context).push(MaterialPageRoute(builder:
 (BuildContext context) {
            return UICollectionHandler();
          })),
        ),
        title: Text(

          //if authentic user state isn't null then show the display name as ema
il display name in the appbar
          authNotifier.user != null ? authNotifier.user.displayName + "'s Favour
ite Fruit Collection": "Your Favourite Collection",
        ),
        actions: <Widget>[

          // action button
          FlatButton(
            onPressed: (){
              Navigator.of(context).push(
                MaterialPageRoute(
                  builder: (BuildContext context){
                    authNotifier.setUser(null);
                    return LoginPageDisplayUI();
                  },
                ),
```

```
              );
            },
            child: Text(
              "Logout",
              style: TextStyle(fontSize: 22, color: Colors.lightBlue),
            ),
          ),
        ],
      ),
      body: new RefreshIndicator(
        child: ListView.separated(
          itemBuilder: (BuildContext context, int index) {
            return ListTile(
              leading: CircleAvatar(
                radius: 25.0,
                child: Image.network(
                  fruitNotifier.fruitList[index].image != null
                      ? fruitNotifier.fruitList[index].image
                      : 'https://www.testingxperts.com/wp-
content/uploads/2019/02/placeholder-img.jpg',
                  width: 150,height: 1000,
                  fit: BoxFit.fill,
                ),
              ),
              title: Text(fruitNotifier.fruitList[index].name),
              subtitle: Text(fruitNotifier.fruitList[index].category),
              onTap: () {
                fruitNotifier.currentFruit = fruitNotifier.fruitList[index];
                Navigator.of(context).push(MaterialPageRoute(builder: (BuildCont
ext context) {
                  return FavouriteDetails();
                }));
              },
            );
          },
          itemCount: fruitNotifier.fruitList.length,
          separatorBuilder: (BuildContext context, int index) {
            InkWell(
              child: ListView(
                shrinkWrap: true,
                primary: false,
                children: <Widget>[
                  Stack(
                    children: <Widget>[
                      Positioned(
                        right: -10.0,
                        bottom: 3.0,
                        child: RawMaterialButton(
```

```
                        onPressed: (){},
                        fillColor: Colors.blue,
                        shape: CircleBorder(),
                        elevation: 4.0,
                        child: Padding(
                          padding: EdgeInsets.all(5),
                          child: Icon(
                            Icons.favorite,
                            color: Colors.red,
                            size: 17,
                          ),
                        ),
                      ),
                    ),
                  ],
                ),
              ],
            ),
            onTap: (){
              Navigator.of(context).push(
                MaterialPageRoute(
                  builder: (BuildContext context){
                    return ProductDetails();
                  },
                ),
              );
            },
          );

          return Divider(
            color: Colors.black,
          );
        },
      ),
      onRefresh: _refreshList,
    ),
  );
 }
}
```

Favourite_Form.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the input screen for fruit crud
Reference-1 : https://github.com/JulianCurrie/CwC_Flutter
```

```
Reference-2 : https://www.youtube.com/watch?v=bjMw89L61FI
Reference-3 : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4 : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
 */

import 'dart:io';

import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import 'package:finalproject/Crudmodel/FruitCrudModel.dart';
import 'package:finalproject/CrudControllers/Fruit_Controller.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:provider/provider.dart';

//creating the class to input favorite fruit details for adding the details of t
he favorite fruit
class FavouriteForm extends StatefulWidget {

  //checking whether needs to update or not
  final bool isUpdating;

  //loading the details to the constructor
  FavouriteForm({@required this.isUpdating});

  //handle the fruit form state
  @override
  _FavouriteFormState createState() => _FavouriteFormState();
}

//creating the class to input favorite fruit details for adding the details of t
he favorite fruit
class _FavouriteFormState extends State<FavouriteForm> {

  //declaring the global form key to maintain form state
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  //declaring the global scaffold key to maintain scaffold state
  final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();

  //creating the list handler to store countries
  List _countries = [];

  //creating the current favorite fruit details object
  FruitCrudModel _currentFruit;

  //creating the image url for the favorite fruit
  String _imageUrl;
```

```
  //creating the image file to store in cloud store bucket
  File _imageFile;

  //creating the text field controller to country adder
  TextEditingController countriesController = new TextEditingController();

  //maintaining the state of the favorite fruit controller
  @override
  void initState() {
    super.initState();

    FruitController fruitNotifier = Provider.of<FruitController>(context, listen
: false);

    //checking if current favorite fruit object is null then load the current de
tails
    if (fruitNotifier.currentFruit != null) {
      _currentFruit = fruitNotifier.currentFruit;
    } else {
      _currentFruit = FruitCrudModel();
    }

    _countries.addAll(_currentFruit.countries);
    _imageUrl = _currentFruit.image;
  }

  //check if the the file and image url is null then show in the placeholder as
image placeholder otherwise print in console as showing image from local file
  _showImage() {
    if (_imageFile == null && _imageUrl == null) {
      return Text(" ");
    } else if (_imageFile != null) {
      print('showing image from local file');

      //designing the ui level
      return Stack(
        alignment: AlignmentDirectional.bottomCenter,
        children: <Widget>[
          Image.file(
            _imageFile,
            fit: BoxFit.cover,
            height: 250,
          ),
          FlatButton(
            padding: EdgeInsets.all(16),
            color: Colors.black54,
            child: Text(
```

```
              'Change Image',
              style: TextStyle(color: Colors.white, fontSize: 22, fontWeight: Fo
ntWeight.w400),
            ),
            onPressed: () => _getLocalImage(),
          )
        ],
      );
    } else if (_imageUrl != null) {
      print('showing image from url');

      return Stack(
        alignment: AlignmentDirectional.bottomCenter,
        children: <Widget>[
          Image.network(
            _imageUrl,
            width: MediaQuery.of(context).size.width,
            fit: BoxFit.fill,
            height: 200,
          ),
          FlatButton(
            padding: EdgeInsets.all(16),
            color: Colors.black54,
            child: Text(
              'Change Image',
              style: TextStyle(color: Colors.white, fontSize: 22, fontWeight: Fo
ntWeight.w400),
            ),
            onPressed: () => _getLocalImage(),
          )
        ],
      );
    }
  }

  //getting the image file from image gallery to load image in the placeholder
  _getLocalImage() async {
    File imageFile =
    await ImagePicker.pickImage(source: ImageSource.gallery, imageQuality: 100,
maxWidth: 400);

    if (imageFile != null) {
      setState(() {
        _imageFile = imageFile;
      });
    }
  }
```

```
  //handle the name field with validation using text form controller
  Widget _buildNameField() {
    return TextFormField(
      decoration: InputDecoration(labelText: 'Fruit Name'),
      initialValue: _currentFruit.name,
      keyboardType: TextInputType.text,
      style: TextStyle(fontSize: 20),
      validator: (String value) {
        if (value.isEmpty) {
          return 'Name is required';
        }

        return null;
      },
      onSaved: (String value) {
        _currentFruit.name = value;
      },
    );
  }

  Widget _buildCategoryField() {
    return TextFormField(
      decoration: InputDecoration(labelText: 'Fruit Family'),
      initialValue: _currentFruit.category,
      keyboardType: TextInputType.text,
      style: TextStyle(fontSize: 20),
      validator: (String value) {
        if (value.isEmpty) {
          return 'Fruit Family is required';
        }

        return null;
      },
      onSaved: (String value) {
        _currentFruit.category = value;
      },
    );
  }

  //handle the available country field with validation using text form controlle
r
  _buildCountryField() {
    return SizedBox(
      width: 200,
      child: TextField(
        controller: countriesController,
        keyboardType: TextInputType.text,
        decoration: InputDecoration(labelText: 'Available Countries'),
```

```
        style: TextStyle(fontSize: 20),
      ),
    );
  }

  //sending the uploaded favorite fruit details to add to the cloud store
  _onFruitUploaded(FruitCrudModel fruit) {
    FruitController fruitNotifier = Provider.of<FruitController>(context, listen
: false);
    fruitNotifier.addFruit(fruit);
    Navigator.pop(context);
  }

  //controlling the text form state whether empty or add the text if it is not e
mpty
  _addCountry(String text) {
    if (text.isNotEmpty) {
      setState(() {
        _countries.add(text);
      });
      countriesController.clear();
    }
  }

   //save the current state of the form text forms
  _saveFavouriteFruit() {
    print('saveFruit Called');
    if (!_formKey.currentState.validate()) {
      return;
    }

    _formKey.currentState.save();

    print('form saved');

    _currentFruit.countries = _countries;

    uploadFavouriteFruitAndImage(_currentFruit, widget.isUpdating, _imageFile, _
onFruitUploaded);

    print("name: ${_currentFruit.name}");
    print("category: ${_currentFruit.category}");
    print("Countries: ${_currentFruit.countries.toString()}");
    print("_imageFile ${_imageFile.toString()}");
    print("_imageUrl $_imageUrl");
  }

  @override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    key: _scaffoldKey,
    appBar: AppBar(title: Text('Favourite Fruit Form')),
    body: SingleChildScrollView(
      padding: EdgeInsets.all(32),
      child: Form(
        key: _formKey,
        autovalidate: true,
        child: Column(children: <Widget>[
          _showImage(),
          SizedBox(height: 16),
          Text(
            widget.isUpdating ? "Add Favourite Fruit" : "Create Fruit",
            textAlign: TextAlign.center,
            style: TextStyle(fontSize: 30),
          ),
          SizedBox(height: 16),
          _imageFile == null && _imageUrl == null
              ? ButtonTheme(
            child: RaisedButton(
              onPressed: () => _getLocalImage(),
              child: Text(
                'Add Image',
                style: TextStyle(color: Colors.white),
              ),
            ),
          )
              : SizedBox(height: 0),
          _buildNameField(),
          _buildCategoryField(),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: <Widget>[
              _buildCountryField(),
              ButtonTheme(
                child: RaisedButton(
                  child: Text('Add', style: TextStyle(color: Colors.white)),
                  onPressed: () => _addCountry(countriesController.text),
                ),
              )
            ],
          ),
          SizedBox(height: 16),
          GridView.count(
            shrinkWrap: true,
            scrollDirection: Axis.vertical,
            padding: EdgeInsets.all(8),
```

```
              crossAxisCount: 3,
              crossAxisSpacing: 4,
              mainAxisSpacing: 4,
              children: _countries
                  .map(
                    (ingredient) => Card(
                  color: Colors.black54,
                  child: Center(
                    child: Text(
                      ingredient,
                      style: TextStyle(color: Colors.white, fontSize: 14),
                    ),
                  ),
                ),
              )
                  .toList(),
            )
          ]),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          FocusScope.of(context).requestFocus(new FocusNode());
          _saveFavouriteFruit();
        },
        child: Icon(Icons.save),
        backgroundColor: Colors.pink,
        foregroundColor: Colors.white,
      ),
    );
  }
}
```

FavouriteDetails.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the favorite fruit detail class show details of the each
favorite fruits
Reference-1 : https://github.com/JulianCurrie/CwC_Flutter
Reference-2 : https://www.youtube.com/watch?v=bjMw89L61FI
Reference-3 : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4 : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
 */
```

```dart
import 'package:finalproject/LearnAFruit_Api/Fruit_Api_Handler.dart';
import 'package:finalproject/Crudmodel/FruitCrudModel.dart';
import 'package:finalproject/CrudControllers/Fruit_Controller.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'dart:io';

//creating the class to display the favorite fruit details from each favorite fr
uits with update delete adding favorite list actions in fruit main screen
class FavouriteDetails extends StatelessWidget {

  //checking whether needs to update or not
  final bool isUpdating;

  //loading the updateble details to the constructor
  FavouriteDetails({@required this.isUpdating});

  //declaring the global scaffold key to maintain scaffold state
  final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();

  //creating the text controller in form section to add new countries to the lis
t of the each favorite fruit
  TextEditingController countriesController = new TextEditingController();

  @override
  Widget build(BuildContext context) {

    //calling the favorite fruit controller class
    FruitController fruitNotifier = Provider.of<FruitController>(context);

    //calling the delete favorite fruit method when invoke the onfruitdeleted me
thod
    _onFruitDeleted(FruitCrudModel fruit) {
      Navigator.pop(context);
      fruitNotifier.deleteFavouriteFruit(fruit);
    }

    //design the ui level
    return Scaffold(
      key: _scaffoldKey,
      appBar: AppBar(
        title: Text(fruitNotifier.currentFruit.name),
      ),
      body: SingleChildScrollView(
        child: Center(
          child: Container(
            child: Column(
              children: <Widget>[
```

```
                //showing the relevant favorite fruit image for each favorite fr
uit detail page or providing to enter new
                Image.network(
                  fruitNotifier.currentFruit.image != null
                      ? fruitNotifier.currentFruit.image
                      : 'https://www.testingxperts.com/wp-
content/uploads/2019/02/placeholder-img.jpg',
                  width: MediaQuery.of(context).size.width,
                  height: 250,
                  fit: BoxFit.fitWidth,
                ),
                SizedBox(height: 24),

                //showing the relevant favorite fruit name for each favorite fru
it detail page or providing to enter new
                Text(
                  fruitNotifier.currentFruit.name,
                  style: TextStyle(
                    fontSize: 40,
                  ),
                ),

                //showing the relevant favorite fruit family name for each favor
ite fruit detail page or providing to enter new
                Text(
                  'Fruit Family: ${fruitNotifier.currentFruit.category}',
                  style: TextStyle(fontSize: 18, fontStyle: FontStyle.italic),
                ),
                SizedBox(height: 20),

                //showing the relevant favorite fruit multiple countries for eac
h favorite fruit detail page  to enter new
                Text(
                  "Available Countries",
                  style: TextStyle(fontSize: 18, decoration: TextDecoration.unde
rline),
                ),
                SizedBox(height: 16),

                //showing the added countries in the list before update
                GridView.count(
                  shrinkWrap: true,
                  scrollDirection: Axis.vertical,
                  padding: EdgeInsets.all(8),
                  crossAxisCount: 3,
                  crossAxisSpacing: 4,
                  mainAxisSpacing: 4,
```

```
                  children: fruitNotifier.currentFruit.countries
                      .map(
                        (ingredient) => Card(
                      color: Colors.black54,
                      child: Center(
                        child: Text(
                          ingredient,
                          style: TextStyle(color: Colors.white, fontSize: 16),
                        ),
                      ),
                    ),
                  )
                      .toList(),
                )
              ],
            ),
          ),
        ),
      ),
      floatingActionButton: Column(
        mainAxisAlignment: MainAxisAlignment.end,
        children: <Widget>[
          SizedBox(height: 20),

          //page button action  to the delete details of the relevant favorite f
ruit
          FloatingActionButton(
            heroTag: 'button2',
            onPressed: () => deleteFruit(fruitNotifier.currentFruit, _onFruitDel
eted),
            child: Icon(Icons.delete),
            backgroundColor: Colors.red,
            foregroundColor: Colors.white,
          ),
        ],
      ),
    );
  }
}
```

LearnaFruitSplashScreen.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference-1 : https://apkpure.com/flutter-mobile-restaurant-
```

```
ui-kit/com.jideguru.restaurant_ui_kit
*/

import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'UICollectionHandler.dart';

class LearnAFruitSplashScreen extends StatefulWidget {
  @override
  _LearnAFruitSplashScreen createState() => _LearnAFruitSplashScreen();
}

class _LearnAFruitSplashScreen extends State<LearnAFruitSplashScreen> {

//set timer to the splash screen
  startTimeout() {
    return  Timer(Duration(seconds: 10), changeScreen);
  }

  //changing the screen into home page after timer is up
  changeScreen() async{
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (BuildContext context){
          return UICollectionHandler();
        },
      ),
    );
  }

  @override
  void initState() {
    super.initState();
    SystemChrome.setEnabledSystemUIOverlays([]);
    startTimeout();
  }

  //UI layout
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).primaryColor,
      body: Container(
        margin: EdgeInsets.only(left: 40.0, right: 40.0, top: 100.0),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[

              //loading the .gif file from assets folder
              Image.asset(
                'assets/learnafruit.gif',
                width: 600.0,
                height: 400.0,
                fit: BoxFit.fitHeight,
              ),
              Padding(
                padding: EdgeInsets.only(top: 40),
              ),
              Expanded(
                flex: 10,
                child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                  children: <Widget>[
                    LinearProgressIndicator(),
                    Padding(
                      padding: EdgeInsets.only(top: 50),
                    ),
                    Text(
                      "Loading Learn a Fruit...",
                      style: TextStyle(color: Colors.green, fontSize: 20),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Fruit_Api_Handler.dart

```
/*
Authors       : IT17136402 - W.M.H.B. Warnakulasooriya, W.G.M.V.S Wijesundara -
 IT17035118
Description   : Creating the Crud Functions
Reference-1   : https://github.com/JulianCurrie/CwC_Flutter
Reference-2   : https://www.youtube.com/watch?v=bjMw89L61FI
```

```
Reference-3   : https://github.com/TechieBlossom/sidebar_animation_flutter
Reference-4   : https://apkpure.com/flutter-mobile-restaurantui-
kit/com.jideguru.restaurant_ui_kit
 */

import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:finalproject/Crudmodel/FruitCrudModel.dart';
import 'package:finalproject/Crudmodel/UserCrudModel.dart';
import 'package:finalproject/CrudControllers/authentication_Controller.dart';
import 'package:finalproject/CrudControllers/Fruit_Controller.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:path/path.dart' as path;
import 'package:uuid/uuid.dart';


login(UserCrudModel user, AuthenticationController authNotifier) async {
  AuthResult authResult = await FirebaseAuth.instance
      .signInWithEmailAndPassword(email: user.email, password: user.password)
      .catchError((error) => print(error.code));

  if (authResult != null) {
    FirebaseUser firebaseUser = authResult.user;

    if (firebaseUser != null) {
      print("Log In: $firebaseUser");
      authNotifier.setUser(firebaseUser);
    }
  }
}

signup(UserCrudModel user, AuthenticationController authNotifier) async {
  AuthResult authResult = await FirebaseAuth.instance
      .createUserWithEmailAndPassword(email: user.email, password: user.password
)
      .catchError((error) => print(error.code));

  if (authResult != null) {
    UserUpdateInfo updateInfo = UserUpdateInfo();
    updateInfo.displayName = user.displayName;

    FirebaseUser firebaseUser = authResult.user;

    if (firebaseUser != null) {
      await firebaseUser.updateProfile(updateInfo);

      await firebaseUser.reload();
```

```
      print("Sign up: $firebaseUser");

      FirebaseUser currentUser = await FirebaseAuth.instance.currentUser();
      authNotifier.setUser(currentUser);
    }
  }
}

signout(AuthenticationController authNotifier) async {
  await FirebaseAuth.instance.signOut().catchError((error) => print(error.code))
;

  authNotifier.setUser(null);
}

initializeCurrentUser(AuthenticationController authNotifier) async {
  FirebaseUser firebaseUser = await FirebaseAuth.instance.currentUser();

  if (firebaseUser != null) {
    print(firebaseUser);
    authNotifier.setUser(firebaseUser);
  }
}

//getting the fruits list from the Fruits Collection in cloud store
getFruits(FruitController fruitNotifier) async {
  QuerySnapshot snapshot = await Firestore.instance
      .collection('Fruits')
      .orderBy("createdAt", descending: true)
      .getDocuments();

  List<FruitCrudModel> _fruitList = [];

  snapshot.documents.forEach((document) {
    FruitCrudModel fruit = FruitCrudModel.fromMap(document.data);
    _fruitList.add(fruit);
  });

  fruitNotifier.fruitList = _fruitList;
}

//getting the favorite fruits list from the FFruits Collection in cloud store
getFavouriteFruits(FruitController fruitNotifier) async {

  //getting the FFruits collections in decending order of created time
  QuerySnapshot snapshot = await Firestore.instance
      .collection('FFruits')
      .orderBy("createdAt", descending: true)
```

```
      .getDocuments();

  //store in list the details of the fruits
  List<FruitCrudModel> _fruitList = [];

 //adding each fruits into fruit list
  snapshot.documents.forEach((document) {
    FruitCrudModel fruit = FruitCrudModel.fromMap(document.data);
    _fruitList.add(fruit);
  });

  fruitNotifier.fruitList = _fruitList;
}

//upload the image into fruits / images bucket as file format
uploadFruitAndImage(FruitCrudModel fruit, bool isUpdating, File localFile, Funct
ion fruitUploaded) async {

  //check if local file is null in the image bucket
  if (localFile != null) {
    print("uploading image");

    //get the path of the image
    var fileExtension = path.extension(localFile.path);
    print(fileExtension);

    //getting the uuid format to store image on buckets
    var uuid = Uuid().v4();

    //refer the storage in fruits/images
    final StorageReference firebaseStorageRef =
        FirebaseStorage.instance.ref().child('fruits/images/$uuid$fileExtension'
);

  //send to the relevant directory in bucket or catch error
    await firebaseStorageRef.putFile(localFile).onComplete.catchError((onError)
{
      print(onError);
      return false;
    });

    //get the image url then assign ito url
    String url = await firebaseStorageRef.getDownloadURL();

    //print url
    print("download url: $url");
```

```
    //then invoke the image url method below to add the url in document collecti
on
    _uploadFruit(fruit, isUpdating, fruitUploaded, imageUrl: url);
  } else {
    print('...skipping image upload');
    _uploadFruit(fruit, isUpdating, fruitUploaded);
  }
}

//upload fruit image url into the Fruits document collection in the relevant fru
it id
_uploadFruit(FruitCrudModel fruit, bool isUpdating, Function fruitUploaded, {Str
ing imageUrl}) async {
  CollectionReference fruitRef = Firestore.instance.collection('Fruits');

  if (imageUrl != null) {
    fruit.image = imageUrl;
  }

  if (isUpdating) {
    fruit.updatedAt = Timestamp.now();

    await fruitRef.document(fruit.id).updateData(fruit.toMap());

    fruitUploaded(fruit);
    print('updated fruit with id: ${fruit.id}');
  } else {
    fruit.createdAt = Timestamp.now();

    DocumentReference documentRef = await fruitRef.add(fruit.toMap());

    fruit.id = documentRef.documentID;

    print('uploaded fruit successfully: ${fruit.toString()}');

    await documentRef.setData(fruit.toMap(), merge: true);

    fruitUploaded(fruit);
  }
}

//upload favourite fruit image url into the FFruits document collection in the r
elevant fruit id
uploadFavouriteFruitAndImage(FruitCrudModel fruit, bool isUpdating, File localFi
le, Function fruitUploaded) async {

  //check if local file is null in the image bucket
  if (localFile != null) {
```

```
    print("uploading image");

    //get the path of the image
    var fileExtension = path.extension(localFile.path);
     print(fileExtension);

    //getting the uuid format to store image on buckets
     var uuid = Uuid().v4();

    //refer the storage in ffruits/images
     final StorageReference firebaseStorageRef =
     FirebaseStorage.instance.ref().child('ffruits/images/$uuid$fileExtension');

    //send to the relevant directory in bucket or catch error
    await firebaseStorageRef.putFile(localFile).onComplete.catchError((onError) {
       print(onError);
       return false;
    });

    //get the image url then assign ito url
     String url = await firebaseStorageRef.getDownloadURL();

    //print url
     print("download url: $url");_uploadFavouriteFruit(fruit, isUpdating, fruitUp
loaded, imageUrl: url);
     print('...skipping image upload');

     //invoke upload favourite fruit to add the image url in document collection
     _uploadFavouriteFruit(fruit, isUpdating, fruitUploaded);
  } else {

     //then upload image url in collection and update the existing image in bucke
t
     print('...skipping image upload');
     _uploadFavouriteFruit(fruit, isUpdating, fruitUploaded);
  }
}

_uploadFavouriteFruit(FruitCrudModel fruit, bool isUpdating, Function fruitUploa
ded, {String imageUrl}) async {

  //refer to the FFruits Collection
  CollectionReference fruitRef = Firestore.instance.collection('FFruits');

//check image url is null
  if (imageUrl != null) {

     //add fruit image url
```

```
    fruit.image = imageUrl;
  }

  //if updating the existing
  if (isUpdating) {

    //create time for created time
    fruit.createdAt = Timestamp.now();

    //refer for fruit add
    DocumentReference documentRef = await fruitRef.add(fruit.toMap());

    //catch the id of relevant fruit
    fruit.id = documentRef.documentID;

    //print upload success with details
    print('uploaded fruit successfully: ${fruit.toString()}');

   //merger all data in collection
    await documentRef.setData(fruit.toMap(), merge: true);

    //invoke fruit upload method
    fruitUploaded(fruit);

  } else {

    //create time for created time
    fruit.createdAt = Timestamp.now();

    //refer for fruit add
    DocumentReference documentRef = await fruitRef.add(fruit.toMap());

    //catch the id of relevant fruit
    fruit.id = documentRef.documentID;

    //print upload success with details
    print('uploaded fruit successfully: ${fruit.toString()}');

    //merger all data in collection
    await documentRef.setData(fruit.toMap(), merge: true);

    //invoke fruit upload method
    fruitUploaded(fruit);
  }
}

//method to delete fruit from collection
deleteFruit(FruitCrudModel fruit, Function fruitDeleted) async {
```

```
  //check whether fruit image null if not
  if (fruit.image != null) {

    //refer from image bucket
    StorageReference storageReference =
        await FirebaseStorage.instance.getReferenceFromUrl(fruit.image);

  //print path of the image
    print(storageReference.path);

    //delete the image
    await storageReference.delete();

    //print as deleted
    print('image deleted');
  }

  //delete the fruit details from the Fruits collection using relevant id
  await Firestore.instance.collection('Fruits').document(fruit.id).delete();

  //invoke fruit delete method
  fruitDeleted(fruit);
}


//method to delete favorite fruit from collection
deleteFavouriteFruit(FruitCrudModel fruit, Function fruitDeleted) async {

  //check whether fruit image null if not
 if (fruit.image != null) {

   //refer from image bucket
    StorageReference storageReference =
    await FirebaseStorage.instance.getReferenceFromUrl(fruit.image);

    //print path of the image
    print(storageReference.path);

    //delete the image
    await storageReference.delete();

    //print as deleted
    print('image deleted');
  }

 //delete the fruit details from the FFruits collection using relevant id
  await Firestore.instance.collection('FFruits').document(fruit.id).delete();
```

```
 //invoke fruit delete method
  fruitDeleted(fruit);
}

//upload the image into user / images bucket as file format
uploadUserAndImage(UserCrudModel user, bool isUpdating, File localFile, Function
 userUploaded) async {

  //check if local file is null in the image bucket
  if (localFile != null) {
    print("uploading image");


    //get the path of the image
    var fileExtension = path.extension(localFile.path);
    print(fileExtension);

    //getting the uuid format to store image on buckets
    var uuid = Uuid().v4();

    //refer the storage in users/images
    final StorageReference firebaseStorageRef =
    FirebaseStorage.instance.ref().child('users/images/$uuid$fileExtension');

  //send to the relevant directory in bucket or catch error
    await firebaseStorageRef.putFile(localFile).onComplete.catchError((onError)
{
      print(onError);
      return false;
    });

    //get the image url then assign it to url
    String url = await firebaseStorageRef.getDownloadURL();

    //print url
    print("download url: $url");

    //then invoke the image url method below to add the url in document collecti
on
    _uploadUser(user, isUpdating, userUploaded, imageUrl: url);
  } else {
    print('...skipping image upload');
    _uploadUser(user, isUpdating, userUploaded);
  }
}

//upload user image url into the Users document collection in the relevant user
id
```

```
_uploadUser(UserCrudModel user, bool isUpdating, Function userUploaded, {String
imageUrl}) async {
  CollectionReference userRef = Firestore.instance.collection('Users');

  if (imageUrl != null) {
    user.image = imageUrl;
  }

  if (isUpdating) {
    user.updatedAt = Timestamp.now();

    await userRef.document(user.email).updateData(user.toMap());

    userUploaded(user);
    print('updated fruit with id: ${user.email}');
  } else {
    user.createdAt = Timestamp.now();

    DocumentReference documentRef = await userRef.add(user.toMap());

    user.email = documentRef.documentID;

    print('uploaded user successfully: ${user.toString()}');

    await documentRef.setData(user.toMap(), merge: true);

    userUploaded(user);
  }
}
```

Detail.dart

```
/*
Author      : IT17136402 - W.M.H.B. Warnakulasooriya
Description : Creating the main screen to direct to the favorite fruit list
Reference-1 : https://github.com/whatsupcoders/Flutter-ImageUpload
Reference-2 : https://apkpure.com/flutter-mobile-restaurant-
ui-kit/com.jideguru.restaurant_ui_kit
*/

import 'package:flutter/material.dart';

class ProductDetails extends StatefulWidget {
  @override
  _ProductDetailsState createState() => _ProductDetailsState();
}
```

```dart
class _ProductDetailsState extends State<ProductDetails> {
  bool isFav = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        leading: IconButton(
          icon: Icon(
            Icons.keyboard_backspace,
          ),
          onPressed: ()=>Navigator.pop(context),
        ),
        centerTitle: true,
        elevation: 0.0,
        actions: <Widget>[
        ],
      ),

      body: Padding(
        padding: EdgeInsets.fromLTRB(10.0,0,10.0,0),
        child: ListView(
          children: <Widget>[
            SizedBox(height: 10.0),
            Stack(
              children: <Widget>[
                Positioned(
                  right: -10.0,
                  bottom: 3.0,
                  child: RawMaterialButton(
                    onPressed: (){},
                    fillColor: Colors.white,
                    shape: CircleBorder(),
                    elevation: 4.0,
                    child: Padding(
                      padding: EdgeInsets.all(5),
                      child: Icon(
                        isFav
                            ?Icons.favorite
                            :Icons.favorite_border,
                        color: Colors.red,
                        size: 17,
                      ),
                    ),
                  ),
                ),
              ],
```

```
            ),
            SizedBox(height: 10.0),
            Padding(
              padding: EdgeInsets.only(bottom: 5.0, top: 2.0),
              child: Row(
                children: <Widget>[
                ],
              ),
            ),
            SizedBox(height: 10.0),
          ],
        ),
      ),
    );
  }
}
```