ÇUKUROVA UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CEN438
GRADUATION THESIS

FLUTTER MOBILE APPLICATION

By

2019555405

Bilge Kaan YUMAK

Advisor

Assoc. Prof. Mustafa ORAL

June 2022

ADANA

ÇUKUROVA UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CEN438
GRADUATION THESIS

FLUTTER MOBILE APPLICATION

By

2019555405

Bilge Kaan YUMAK

Advisor

Assoc. Prof. Mustafa ORAL

June 2022

ADANA

ABSTRACT

The steps of this project development process were supplied and introduced in the study, along with project visuals. A mobile application was designed to meet the hobbies of a person. Since the requirements of hobbies differ from person to person, a software solution based on general common requirements was proposed. The development process of the graduation project was explained step by step. These stages include database field relations and user role relationship in design, user interface design, and the working principles of the software.

This is a location-based project that offers a digital way to take under record when user sees something remarkable (i.e. an irresistible flower, a gold mine, fishing point of a lake or a thyme growing point on mountain). Project provides taking notes and photos, getting location for record purpose to access when the user needs, list them for draw a route or shows markers(with added note field) of locations on the map to see the place with easy use interface design. The goal of this project is to find a solution that digitizes remembering locations about remarkable stuffs with help of location service, photos and notes way filtering premium or free member's data using Flutter Dart Mobile Development Language, Firebase as database provider, also Firebase User Authentication to match registered user information and Google Maps.

GLOSSARY OF TERMS

A

API
An application programming interface (API) is a connection between computers or between computer programs. 3

C

Cloud FireStore
Sync data across devices, on or offline. 1

D

Database
A structured collection of data that is typically stored and retrieved electronically via a computer system. 4

F

Firebase
Firebase is an application development platform that helps developer to build and grow apps and games 3

I

IDE
Integrated Development Environment 1

N

NoSQL
Stands for Not-Only-SQL. NoSQL database technology stores information in JSON documents instead of columns and rows used by relational databases. 1

O

O.O.P
Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects. 1

S

SDKs
SDK stands for software development kit. Also known as a devkit, the SDK is a set of software-building tools for a specific platform, including the building blocks, debuggers and, often, a framework or group of code libraries such as a set of routines specific to an operating system. 3

U

UI
The "UI" abbreviation stands for "user interface," which refers to the graphical layout of an application. 2

FIGURES

TABLE OF CONTENTS

INTRODUCTION

With today's technological advancements, mobile systems make the life easier. Mobile applications became important elements of life, this graduation thesis is a mobile application development project that stores uploaded taken photo by user, provided description from user, the user's current location when they add the data about the stuff or place. After marking and saving data when users leave the place, they can easily reach and draw a route to marked location from the listed places menu called “Marked Locations”.

Users dividing 2 categories as VIP and Free. VIP users can see all marked locations include each free member’s marks except other VIP users. Free users only can see each other’s marks.

This work employs Google Maps, Cloud FireStore technologies and interacts dynamically with a NoSQL database.

Nowadays with the improvement of technology people searches for application to make their lives easy for complex works or hobbies. This project was developed by considering the average requests of people.

This work consists of research on programming languages, database to be used and application itself. The Dart programming language, which is an mixed O.O.P and Functional, has been researched. There is information regarding the materials and methods that are used.

The Android Studio IDE, which is utilized as a Dart programming language compiler, is analyzed, and the application graphical interface that are provided as visuals. Google Maps and Cloud-Firestore were also analyzed for use in the application, which are the cornerstone of this project. Mostly codes and interfaces in the project is covered, and the connection interfaces in the No-SQL database utilized are provided as visuals.

1.1 Software

Software is a set of machine commands used to improve the functionalities or usability of electronic devices meant to execute a variety of activities by allowing them to communicate and adapt to one another.[4]

1.2 User-Friendly Characteristics

Some rules must be followed in order to make the software both visually appealing and functional.

1.2.1 UI's Must-Have Features

Accuracy: The program should be able to complete the task precisely without error.
Simple to Use: People who will utilize the program should be able to use it without much assistance.
Simplicity: The program's UI should be as straightforward as possible.
Durability: The program should not crash as a result of unanticipated mistakes.
Compatibility: The software should be able to execute and be compatible with higher versions of the system on which it is operating. [5]

1.2.2 Software's Must-Have Features

Extensibility: The program should be open to development and feature expansion.
Modularity: The program should be able to be used by other programs.

Controllability: Descriptive error messages should be posted in areas where errors are likely to occur.

Degradability: The program should be able to be written in small chunks. The program can be created more easily and healthily this way.

Understandability: When another programmer reads written program codes, they can be easily comprehended. Messages detailing what it does should be written next to the codes to accomplish this. [5]

1.3 Flutter: Dart Programming Language

Flutter is Google's portable UI toolkit for natively compiled applications for mobile(Android, IOS), web, and desktop from a single codebase. **Flutter does not provide to developers drag and drop GUI features, so the developer must define by code every each of graphical content.**

Flutter uses Dart as a programming language. Dart is a type-safe, object-oriented programming[2] language that is modern and object-oriented. Dart allows developers to create a widget based, safely designed, multi-functional, object oriented and easy-coded applications. [6]

1.4 Firebase: Cloud Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. It keeps the data in sync across client apps through real-time listeners and offers offline support for mobile and web so developer can build responsive apps that work regardless of network latency or Internet connectivity.

Cloud Firestore is a cloud-hosted, No-SQL database that developer's Apple, Android, and web apps can access directly via native SDKs. [8]

Following Cloud Firestore's NoSQL data model, developer store data in documents that contain fields mapping to values. These documents are stored in collections, which are containers for the documents that developers can use to organize their data and build queries easily like adding, updating, getting, where, deleting, etc.

Firebase provides API for developers to bound their application to this service.

1.5 Firebase: User Authentication

Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all the user's devices.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to app. It supports authentication using passwords, phone numbers, popular federated identity providers like social media platforms [1]

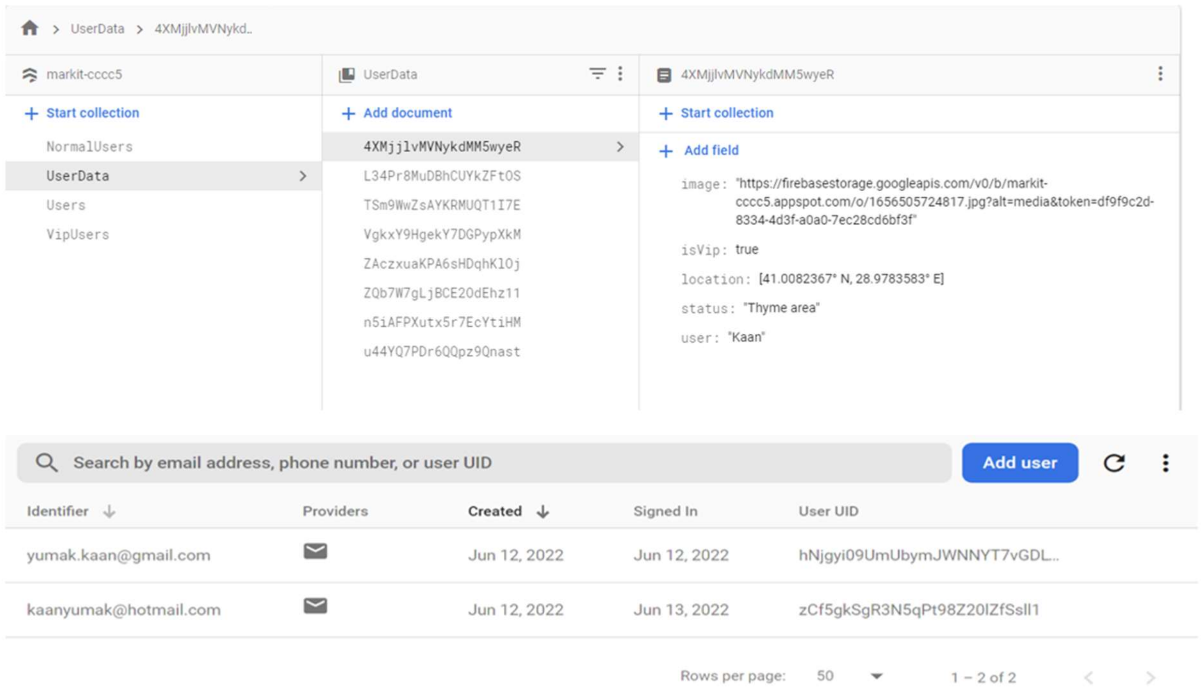
2. DATABASE DESIGN

The development sequence in such projects is the database modeling, UI drawing, and coding. The development order was followed in the same manner in this project.[7]

The collections were established once the needs analysis was completed during the database creation process. 2 collections were created as “Users” and “UserData”. These are consisting of documents which are random-unique named and the documents are consist of data fields.

The document of “Users” stores information about users. “Users” collection store user’s document id by randomly created unique id when the create method calls. This document has fields to keep user's user type as Boolean "isVip" for control the privilege on the data, username as String "userName" and email as String "eMail" parameters for login procedure since the Firestore authentication method "userEmailPasswordLogin" requires this mandatory at least 2-field pattern. After user creation process user's information also will be stored at Firebase Authentication services to check password authentication, identifier, created and last signed dates and to configure account with disable, delete or reset password functions which provided by Firebase.

The document of "UserData" stores information about data. "UserData" collection store user's data document id by randomly created unique id when the add method calls. This document has fields to keep provided data by user's uploaded image URL as String "image", location as Geolocation "Latitude, Longitude", user type as Boolean "isVip", written note by user as String "status", author name as String "user" parameters.



NormalUsers
UserData
Users
VipUsers

4XMjjlvMVNykdMM5wyeR
L34Pr8MuDBhCUYkZFtOS
TSm9WwZsAYKRMUQT1I7E
VgkxY9HgekY7DGPypXkM
ZAczxuaKPA6sHDqhK10j
ZQb7W7gLjBCE20dEhz11
n5iAFPXutx5r7EcYtiHM
u44YQ7PDr6QQpz9Qnast

```
image: "https://firebasestorage.googleapis.com/v0/b/markit-cccc5.appspot.com/o/1656505724817.jpg?alt=media&token=df9f9c2d-8334-4d3f-a0a0-7ec28cd6bf3f"
isVip: true
location: [41.0082367° N, 28.9783583° E]
status: "Thyme area"
user: "Kaan"
```

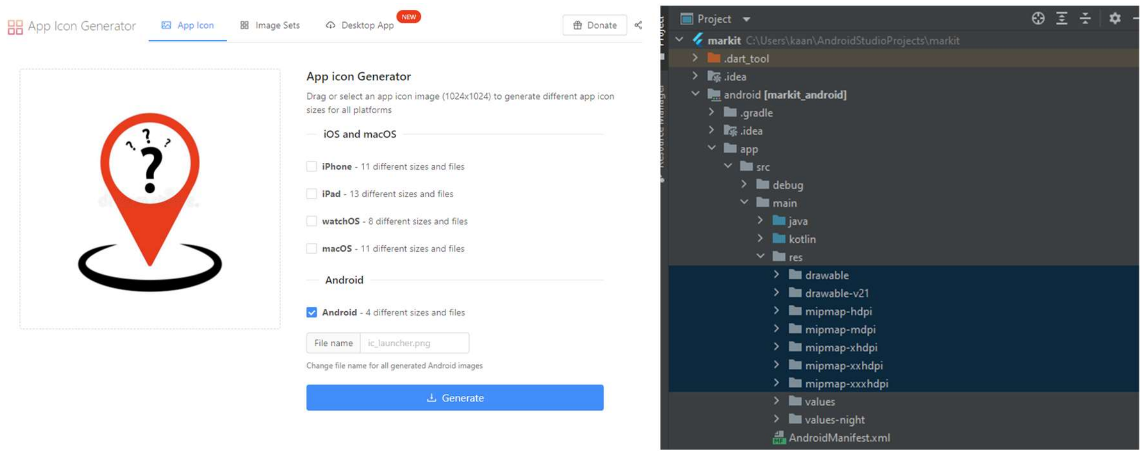
Identifier	Providers	Created	Signed In	User UID
yumak.kaan@gmail.com	✉	Jun 12, 2022	Jun 12, 2022	hNjgyi09UmUbymJWNNYT7vGDL...
kaanyumak@hotmail.com	✉	Jun 12, 2022	Jun 13, 2022	zCf5gkSgR3N5qPt98Z20lZfSsll1

Figure 1: Authentication Data, Database Collections and Documents

3. PAGE DESIGNS

3.1 Application Logo

First, symbol of application on the menu must be decided/designed so that app can be memorable. After that next step is deciding the size. With the help of some websites which are designed to convert uploaded photo various sizes it can be done. In this project "appicon.co" website used for this purpose and generated various type of size as in Figure 2.



App icon Generator

Drag or select an app icon image (1024x1024) to generate different app icon sizes for all platforms

iOS and macOS

- [ ] **iPhone** - 11 different sizes and files
- [ ] **iPad** - 13 different sizes and files
- [ ] **watchOS** - 8 different sizes and files
- [ ] **macOS** - 11 different sizes and files

Android

- [✓] **Android** - 4 different sizes and files

File name: ic_launcher.png

Change file name for all generated Android images

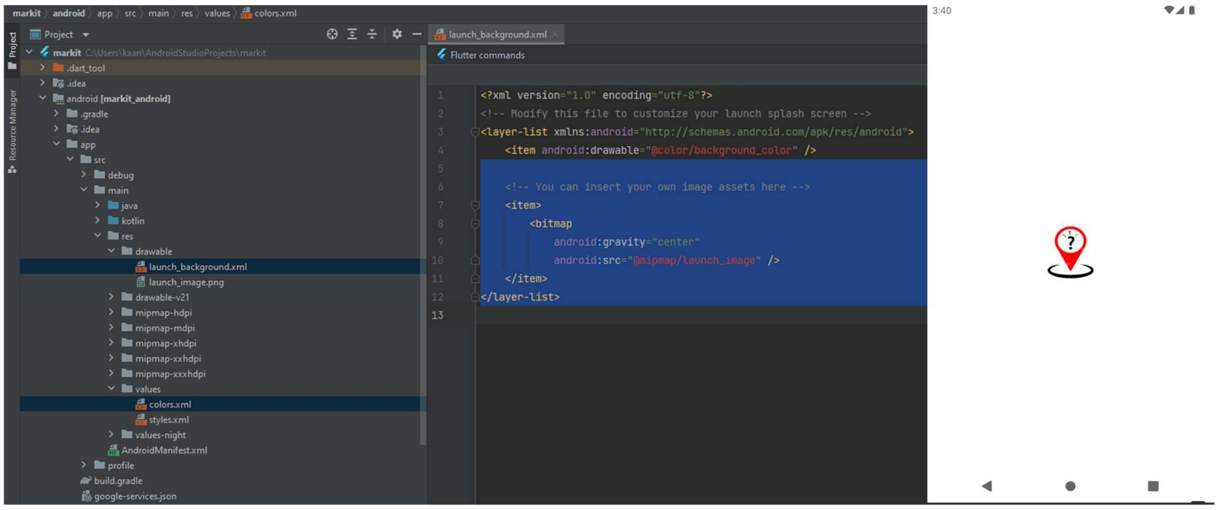
```
Project
markit C:\Users\kaan\AndroidStudioProjects\markit
  .dart_tool
  .idea
  android [markit_android]
    .gradle
    .idea
    app
      src
        debug
        main
          java
          kotlin
          res
            drawable
            drawable-v21
            mipmap-hdpi
            mipmap-mdpi
            mipmap-xhdpi
            mipmap-xxhdpi
            mipmap-xxxhdpi
            values
            values-night
          AndroidManifest.xml
```

Figure 2: Application logo

On the second step as shown in Figure 2, files must be in related project folder>app>src>main>res so that the process could be successfully done.

3.1.1 Using Application Logo as Splash Screen

Created logo need to be located as in the figure 3 under the first selection which is bounded with drawable direction. launch_background.xml file must have changed as below. This xml file fetching color from the colors.xml file and according to the white color it is arranged as #FFFFFF



```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/background_color" />

    <!-- You can insert your own image assets here -->
    <item>
        <bitmap
            android:gravity="center"
            android:src="@mipmap/launch_image" />
    </item>
</layer-list>
```

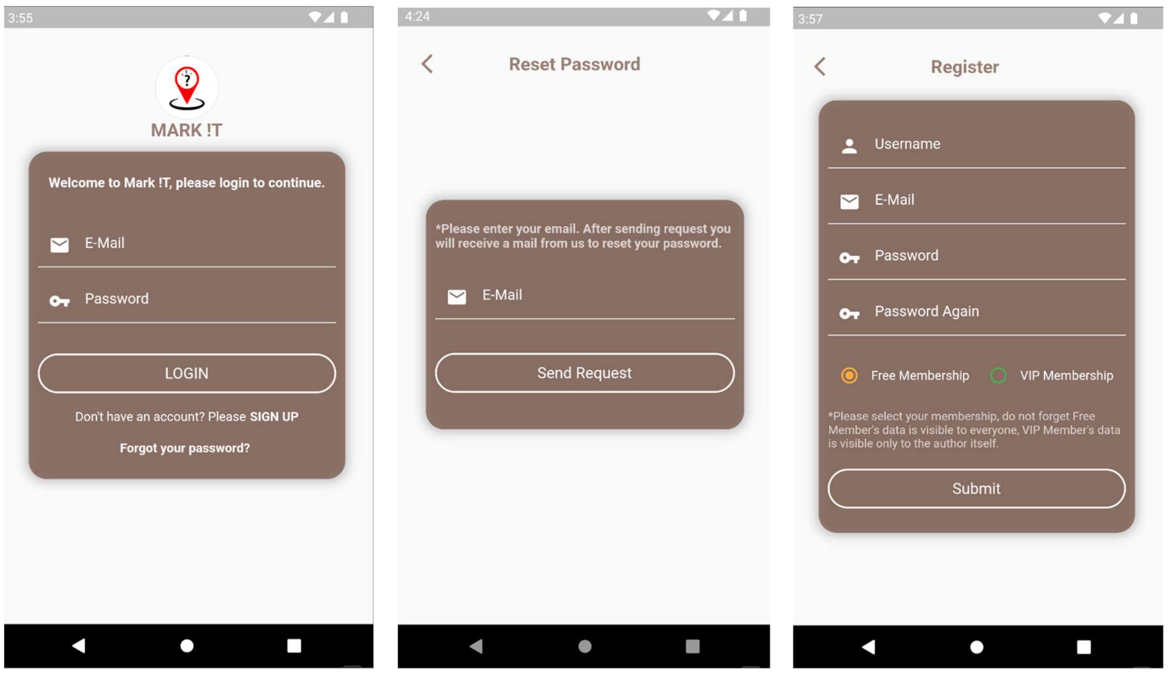
Figure 3: Using Logo as Splash Screen

3.3 Login Page and Registration

Flutter works with widget-based design. It is very similar to div structures in HTML and Scene/Pane in Java FX. First the page structure must be designed according to data type and requirements and then coding part had to be complete. Page designed as in Figure 4 and it takes user's phone screen size by using "var size = MediaQuery.of(context).size;" and creating materials with responsive size by doing multiplication with using this variable(size*0.03 etc). As in the CSS, styling procedures should be done using "style" declaration.

When user runs the application Login Page opens to make the data personal and provide security purpose. [9] In this page there is a logo on the top, app name under the logo, an instruction for the user, controlled e-mail area, hided-typed password area(storing in Firebase with firebase own SCRYP base64 hash algorithm), a login button, sign up button and forgot password options. After clicking login, app establishes a connection with Firebase authentication services and FireStore database to ready to fetch the data.

If the user's does not have an account, then they must click the sign-up button and create an account.



Welcome to Mark !T, please login to continue.

*Please enter your email. After sending request you will receive a mail from us to reset your password.

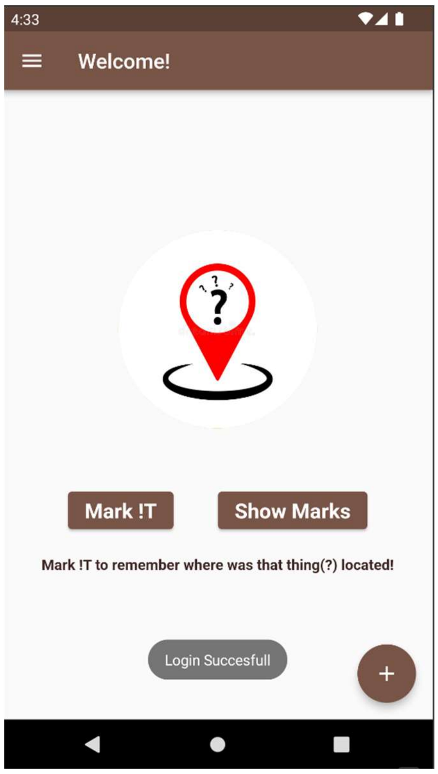
*Please select your membership, do not forget Free Member's data is visible to everyone, VIP Member's data is visible only to the author itself.

Figure 4: Login Page and Registration Design

As seen in the figure 4, register page have places with requirements to sign-up and radio buttons to distinguish user type as Free Member or VIP Member. Users dividing 2 categories as VIP and Free. VIP users can see all marked locations include each free member's marks except other VIP users. Free users only can see each other's marks.

3.4 Home Page and Hidden Menu

In Home Page users first see a bottom located an informative toast message for a few seconds, meets the welcoming page, from here a new data can be entered from Mark !T button also bottom-right located FloatingActionButton which is "+" shaped button, due to user type filtered marked locations query can be made from the database by clicking "Show Marks" button.



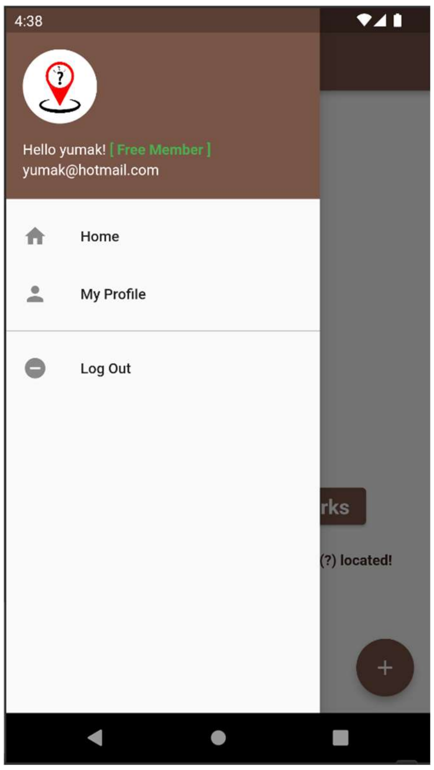


Figure 5: Home Page and Hidden Menu Design

There is a hidden list tile menu top-left on the screen, by clicking this button user can see app logo on the top-left, their logged account with their names along greeting message, e-mail addresses and green colored user types can click home button to turn back this home page on the future, can see their profile information and they may log out.

3.5 Add Mark Page

In this page application collects the note, current location, and photo(which is provided by user's camera device or phone's gallery. If not, default image would be selected to provide Null Safety) from the user and stores in the database's related fields as shown in figure 1.

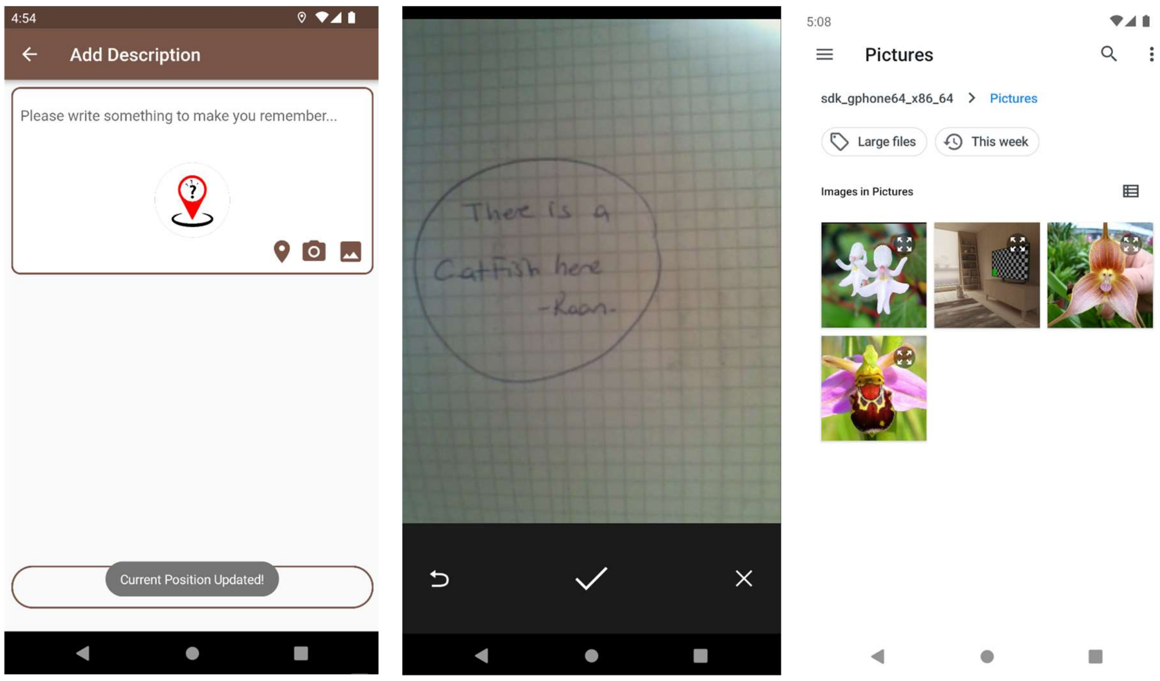
There are four buttons in this page.

- Location-icon fetches the current location,

- Camera-icon give ability to users to upload photo by camera device

- Image-icon, gives access to users to upload photo from gallery.

When users enter a note, upload or takes photo with camera and click add button there is a message shown on the screen as "Data has been added!" and route users to marked locations listed entries.



Please write something to make you remember...

Current Position Updated!

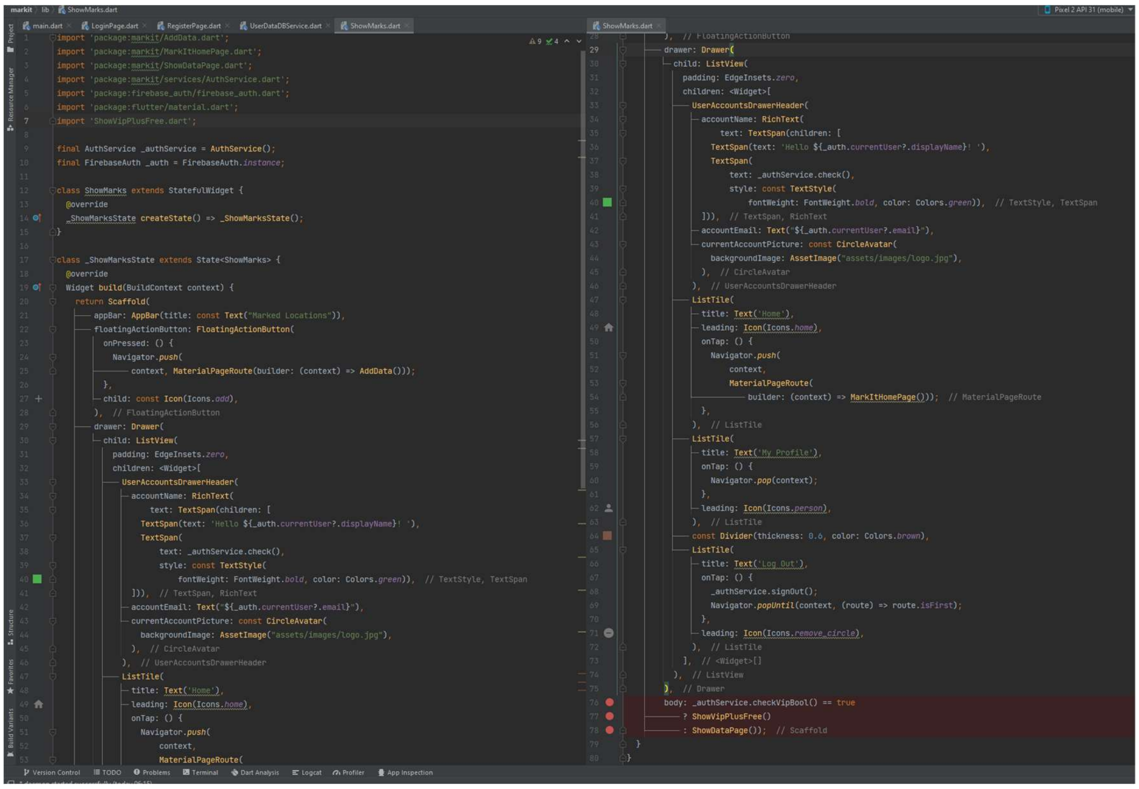
There is a Catfish here -Kaan.

Figure 6: Add Mark Page Design

3.4 Show Marks Page

When the user tapped “Show Marks” button :

Rendering process starts and draw the widgets until the body. On the body part app will enter the if else conditional ternary statement by comparing the current user’s isVip field is true or false as shown in figure 7, red-dot-selected part.



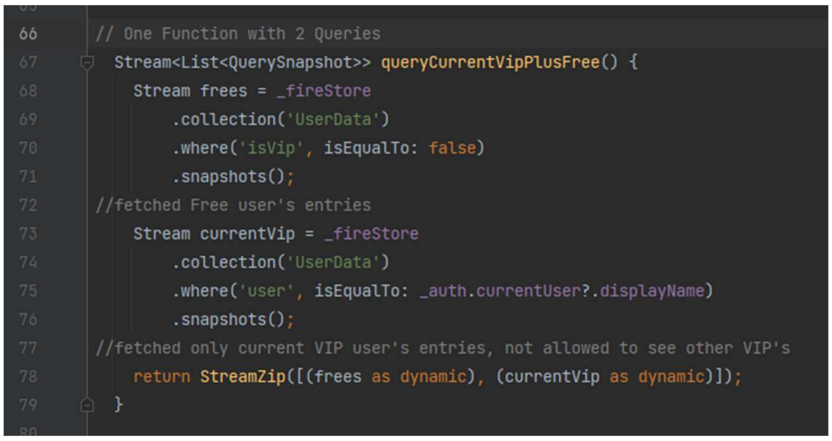
```dart
import 'package:markit/AddData.dart';
import 'package:markit/MarkItHomePage.dart';
import 'package:markit/ShowDataPage.dart';
import 'package:markit/services/AuthService.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'ShowVipPlusFree.dart';

final AuthService _authService = AuthService();
final FirebaseAuth _auth = FirebaseAuth.instance;

class ShowMarks extends StatefulWidget {
  @override
  _ShowMarksState createState() => _ShowMarksState();
}

class _ShowMarksState extends State<ShowMarks> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Marked Locations")),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
              context, MaterialPageRoute(builder: (context) => AddData()));
        },
        child: const Icon(Icons.add),
      ), // FloatingActionButton
      drawer: Drawer(
        child: ListView(
          padding: EdgeInsets.zero,
          children: <Widget>[
            UserAccountsDrawerHeader(
              accountName: RichText(
                  text: TextSpan(children: [
                TextSpan(text: 'Hello ${_auth.currentUser?.displayName}! '),
                TextSpan(
                    text: _authService.check(),
                    style: const TextStyle(
                        fontWeight: FontWeight.bold, color: Colors.green)),  // TextStyle, TextSpan
              ])),  // TextSpan, RichText
              accountEmail: Text("${_auth.currentUser?.email}"),
              currentAccountPicture: const CircleAvatar(
                backgroundImage: AssetImage("assets/images/logo.jpg"),
              ),  // CircleAvatar
            ),  // UserAccountsDrawerHeader
            ListTile(
              title: Text('Home'),
              leading: Icon(Icons.home),
              onTap: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
```

```dart
      ),  // FloatingActionButton
      drawer: Drawer(
        child: ListView(
          padding: EdgeInsets.zero,
          children: <Widget>[
            UserAccountsDrawerHeader(
              accountName: RichText(
                  text: TextSpan(children: [
                TextSpan(text: 'Hello ${_auth.currentUser?.displayName}! '),
                TextSpan(
                    text: _authService.check(),
                    style: const TextStyle(
                        fontWeight: FontWeight.bold, color: Colors.green)),  // TextStyle, TextSpan
              ])),  // TextSpan, RichText
              accountEmail: Text("${_auth.currentUser?.email}"),
              currentAccountPicture: const CircleAvatar(
                backgroundImage: AssetImage("assets/images/logo.jpg"),
              ),  // CircleAvatar
            ),  // UserAccountsDrawerHeader
            ListTile(
              title: Text('Home'),
              leading: Icon(Icons.home),
              onTap: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => MarkItHomePage()));  // MaterialPageRoute
              },
            ),  // ListTile
            ListTile(
              title: Text('My Profile'),
              onTap: () {
                Navigator.pop(context);
              },
              leading: Icon(Icons.person),
            ),  // ListTile
            const Divider(thickness: 0.6, color: Colors.brown),
            ListTile(
              title: Text('Log Out'),
              onTap: () {
                _authService.signOut();
                Navigator.popUntil(context, (route) => route.isFirst);
              },
              leading: Icon(Icons.remove_circle),
            ),  // ListTile
          ],  // <Widget>[]
        ),  // ListView
      ),  // Drawer
      body: _authService.checkVipBool() == true
          ? ShowVipPlusFree()
          : ShowDataPage());  // Scaffold
  }
}
```

Figure 7: User Type Condition Rendering

- If it is true then it will draw rest of it considering the user as VIP member and do this query;

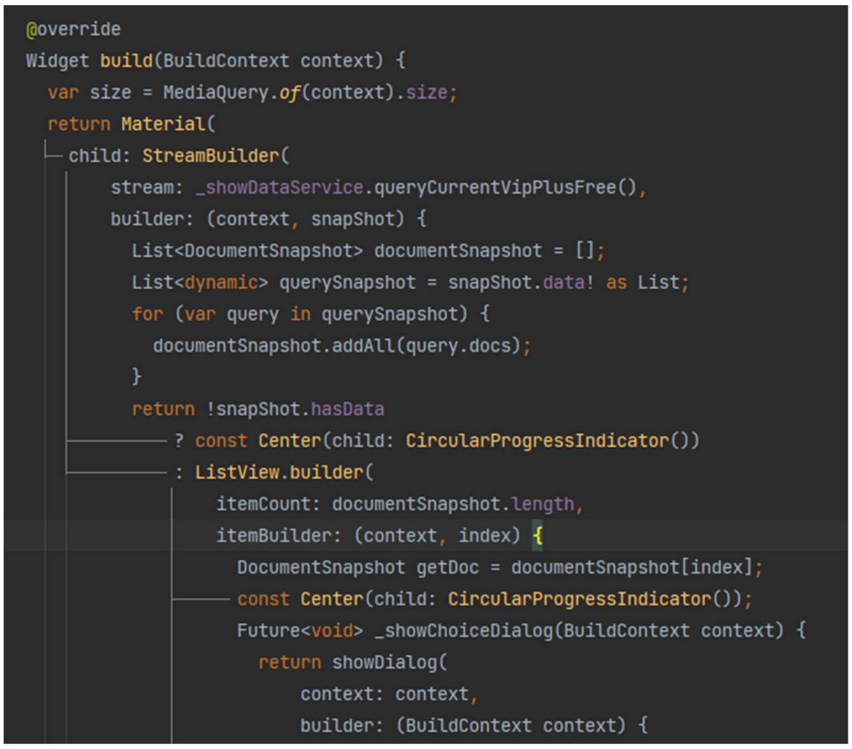
Database query procedure starts(_fireStore variable is a FireBase instance).



```
// One Function with 2 Queries
Stream<List<QuerySnapshot>> queryCurrentVipPlusFree() {
  Stream frees = _fireStore
      .collection('UserData')
      .where('isVip', isEqualTo: false)
      .snapshots();
//fetched Free user's entries
  Stream currentVip = _fireStore
      .collection('UserData')
      .where('user', isEqualTo: _auth.currentUser?.displayName)
      .snapshots();
//fetched only current VIP user's entries, not allowed to see other VIP's
  return StreamZip([(frees as dynamic), (currentVip as dynamic)]);
}
```

Figure 8: Querying Current VIP User and Free User Data

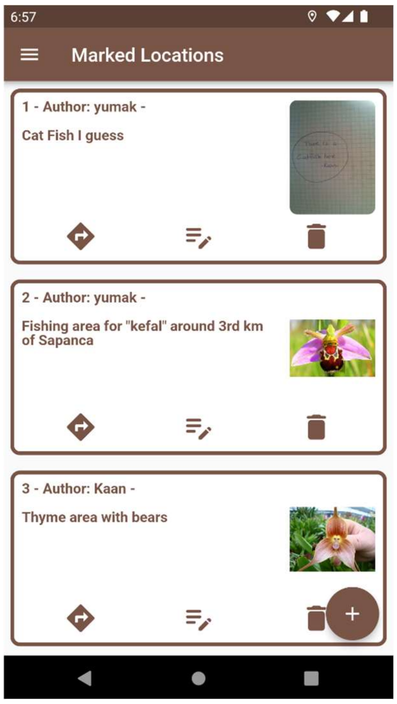
Since two queries at a time restricted and where queries acting as a Logical AND operator between fields, queries completing and adding one by one to Stream(dynamic) List and will again convert again to original object type while listing as below.



```
@override
Widget build(BuildContext context) {
  var size = MediaQuery.of(context).size;
  return Material(
    child: StreamBuilder(
        stream: _showDataService.queryCurrentVipPlusFree(),
        builder: (context, snapShot) {
          List<DocumentSnapshot> documentSnapshot = [];
          List<dynamic> querySnapshot = snapShot.data! as List;
          for (var query in querySnapshot) {
            documentSnapshot.addAll(query.docs);
          }
          return !snapShot.hasData
              ? const Center(child: CircularProgressIndicator())
              : ListView.builder(
                  itemCount: documentSnapshot.length,
                  itemBuilder: (context, index) {
                    DocumentSnapshot getDoc = documentSnapshot[index];
                    const Center(child: CircularProgressIndicator());
                    Future<void> _showChoiceDialog(BuildContext context) {
                      return showDialog(
                          context: context,
                          builder: (BuildContext context) {
```

Figure 9: VIP Type Listing

Due to the queried database, data is retrieved from the database and displayed on the screen as list view. Since the user "Kaan" is VIP, he sees all records except other VIPs.


6:57

Marked Locations

1 - Author: yumak -

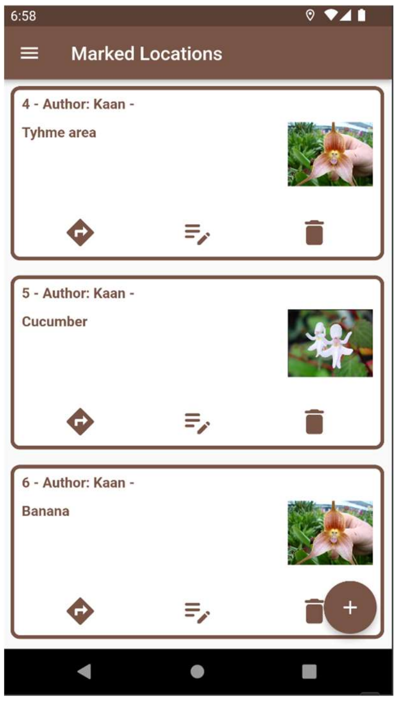
Cat Fish I guess

2 - Author: yumak -

Fishing area for "kefal" around 3rd km of Sapanca

3 - Author: Kaan -

Thyme area with bears


6:58

Marked Locations

4 - Author: Kaan -

Tyhme area

5 - Author: Kaan -

Cucumber

6 - Author: Kaan -

Banana

Figure 10: Show Marks Page Design for VIPs

- If it is false in Figure 7 condition then it will draw rest of the page considering the user as Free member and do this query to list entries;

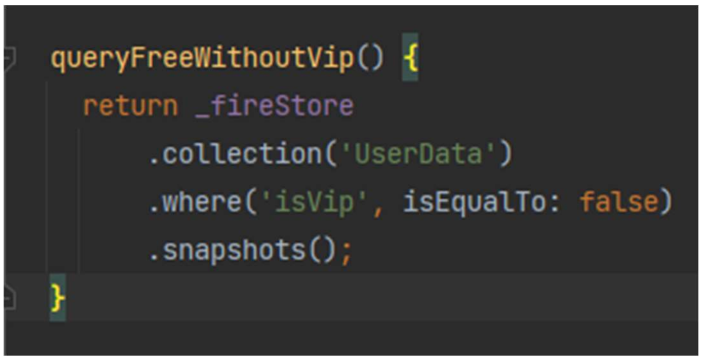
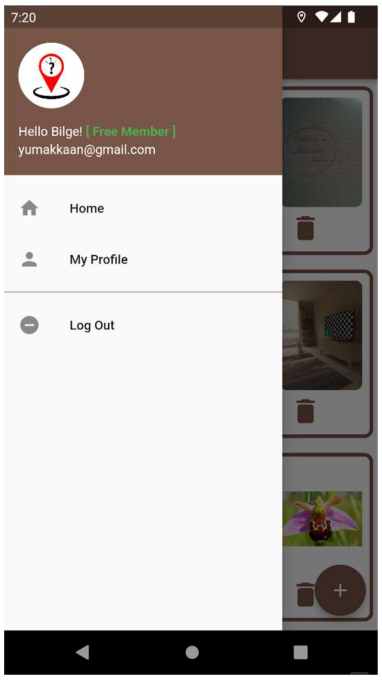

```
queryFreeWithoutVip() {
  return _fireStore
      .collection('UserData')
      .where('isVip', isEqualTo: false)
      .snapshots();
}
```

Figure 11: Querying Free User Data

With same procedures above except converting part(because here conversion is not necessary since there is a one query at a time).

Due to the queried database, data is retrieved from the database and displayed on the screen as list view. Since user changed “Kaan” to “yumak” and another free member “Bilge” is added “Test Data” as an entry. Regarding to condition “yumak” sees only free members records.



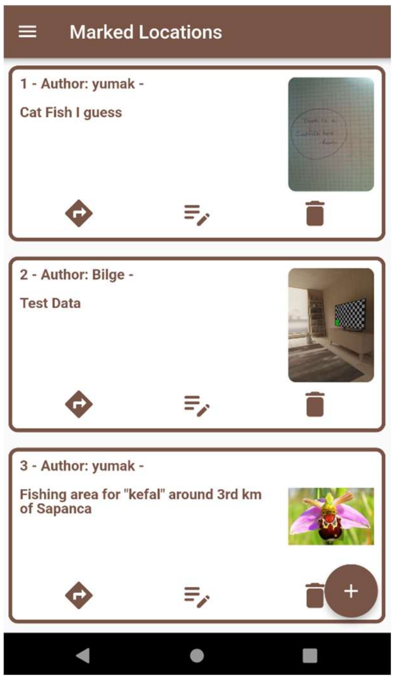
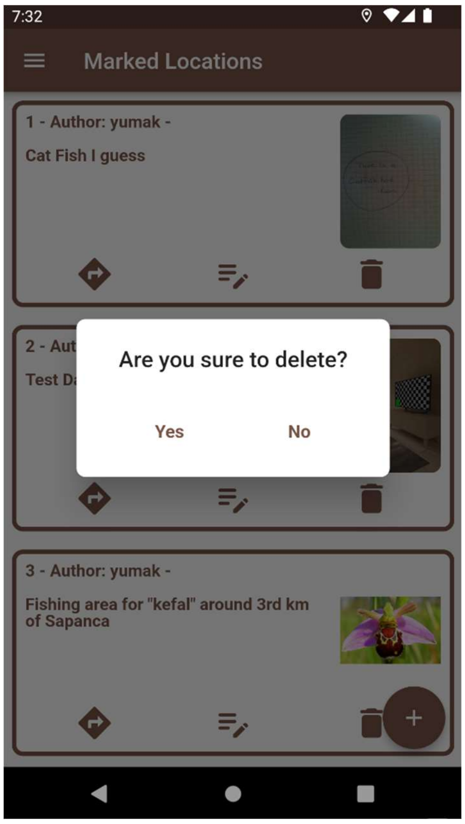


Figure 12: Show Marks Page Design for Free Users

This page provides add new entry, edit description and delete entry from the database, review stored location with description marker on the map and start a route to get directions.

3.4.1 Delete Entry

When users clicked trash-bin shaped button a dialog box pop-up and write a message to the screen “Are you sure to delete it?” If users select the yes answer, it is immediately deleting from the database permanently. Deleting 2nd item from the list shown below in the Figure-13.



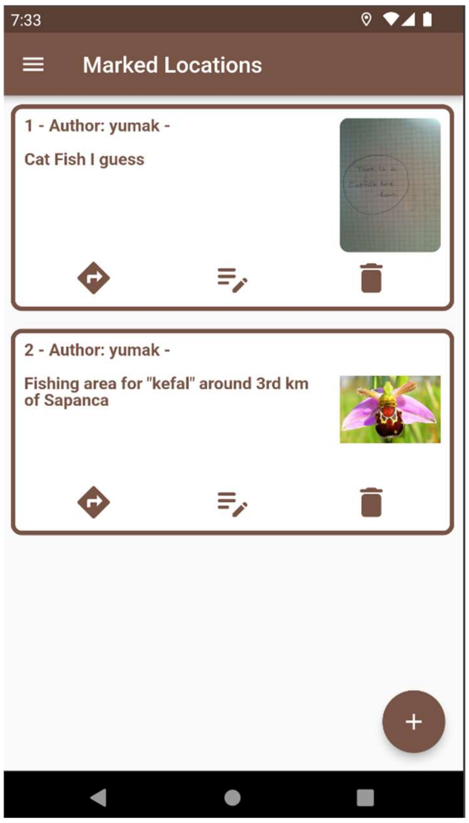


Figure 13: Delete Entry Design

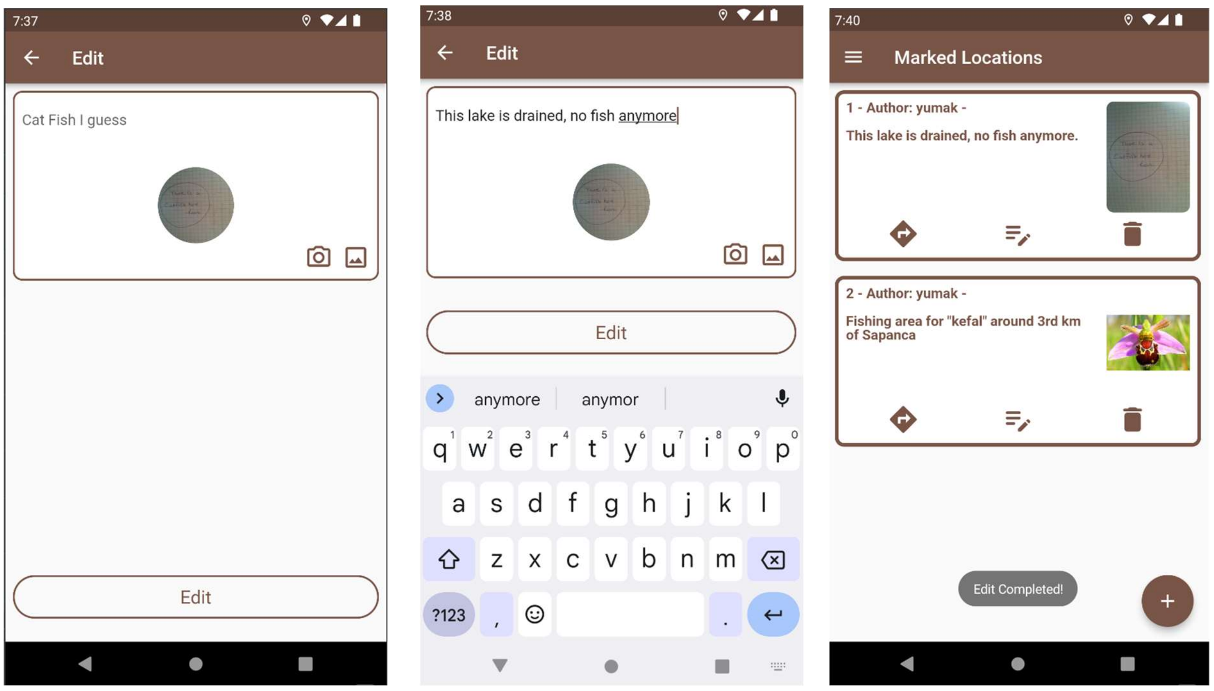
3.4.2 Edit Entry

The notepad-shaped button provides to edit the entry.

The entry which is added by users can be edited with conditions, they cannot change the location and photo to prevent them miss-leading also protect data to possible unintended changes. Users only can change the description part.

As shown in the figure, application fetches old description with low opacity, upload media buttons are outlined. Also there is no location button, means auto-fetch location service is disabled and not updatable.

Edit button updates the entry with new description which related Collection>Document.id>Note(status in database) field with no touching photo field and location field.



Cat Fish I guess

This lake is drained, no fish anymore

1 - Author: yumak -
This lake is drained, no fish anymore.

2 - Author: yumak -
Fishing area for "kefal" around 3rd km of Sapanca

Figure 14:Edit Entry Design

3.4.3 See Location Marks and Get Directions

The traffic sign-shaped button provides users to get directions with drawn route on the map from current location to entry’s stored geological point.

At this point Free member user “bilge” changed to VIP user “Kaan” for accessing all free members also.

Using Google Maps and Firebases need some requirements as taking API key from these providers and match it with developers SHA-1[8]

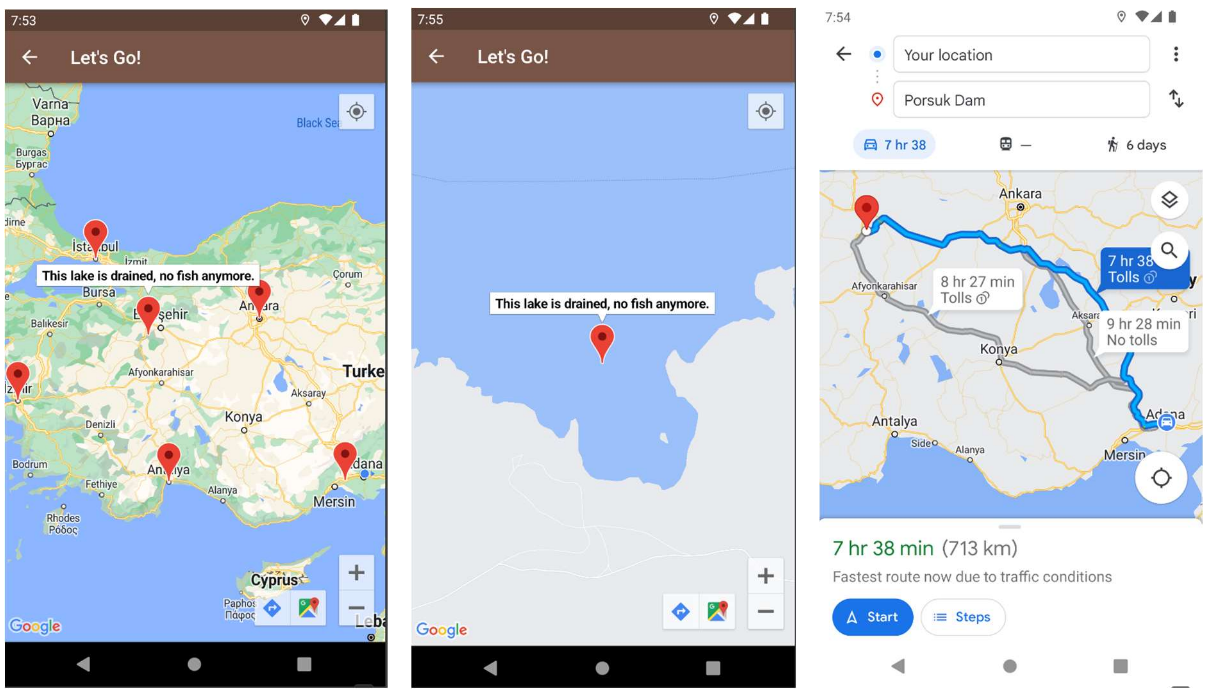
- Google Maps Page starts with where initialCameraPosition equals clicked entry location with 7x zoom. It is very enough to see the relevant entry’s location.
- There are 4 buttons on this page.

Back-arrow located top-left for previous page,

Compass located top right for initialize the camera of the map to current location,

Symbols which represent the “+” and “-“ are zoom in and zoom out,

Red marker(listed all entries according to user rule), when user tap on the marker, information box occurs and retrieve related entry’s description field from database so that another button becomes visible which enables to user get directions.


This lake is drained, no fish anymore.

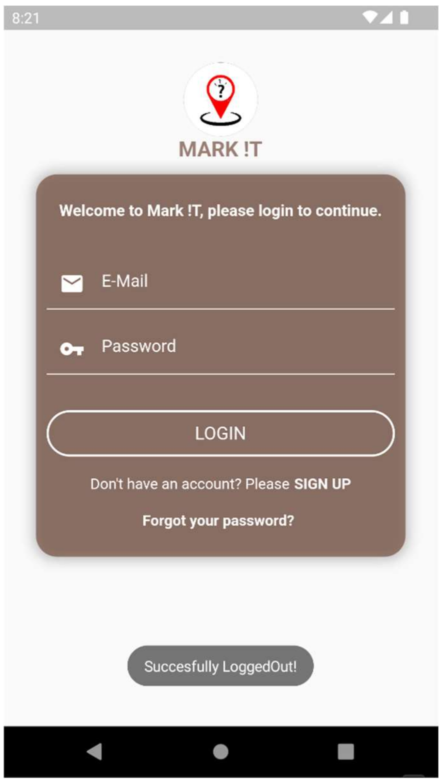
This lake is drained, no fish anymore.

7 hr 38 min (713 km)
Fastest route now due to traffic conditions

Figure 15: Get Directions Page Design

3.5 Log Out

In security reasons or switch between accounts user may log out with clicking on this button.



Welcome to Mark !T, please login to continue.

Don't have an account? Please **SIGN UP**

Forgot your password?

Succesfully LoggedOut!

Figure 16: Log Out

CONCLUSION

In this study, assumed a person sees an irresistible flower(or flowers) and wanted to move it to their backyard and use this application to mark it so that the data entry begins with this point at some place on earth and ends with the stored that flower’s location, photo and remember note. That person can go that location whenever he/she want and have assist able information with finding and remembering it, also when that person goes and find it, they can delete the record or replace the note which is written on the mark. The Mark !T mobile application, which is a common need in for hobbies for people, was realized with this project using Dart, one of the most up-to-date object-based languages, Google Maps and FireStore No-SQL database. The project's database design, page design, and working principle are all shown and explained.

REFERENCES

[1] Firebase,(12 May 2022),What is Firebase :

[https://firebase.google.com/docs/auth]

[2] Wikipedia,(28 May 2022),Object-oriented programming :

[https://en.wikipedia.org/wiki/Object-oriented_programming]

[3] Firebase documentation,(May 2022), What is a No-SQL database? :

[https://firebase.google.com/docs/firestore]

[4] Wikipedia,(04 May 2022),Yazılım :

[https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m#cite_note-1]

[5] Wikibooks, (28 May 2022), Programlamatemelleri/yazılım :

[https://en.wikipedia.org/wi https://en.wikipedia.org/wiki/APIki/API]

[6] Dart.dev, (03 May 2021),Dart overview :

[https://dart.dev/overview]

[7] Flutter,(02 May 2021), Flutter Design, Begin Dart:

[https://docs.flutter.dev/cookbook/design]

[8] How to take an API key,(26 May 2021), Google,

[console.cloud.google.com/apis/support]

[9] pub.dev , ,(02 May 2021), Flutter Examples for Beginners.

[pub.dev]

ACKNOWLEDGEMENTS

I would like to thank my consultant, Assoc. Professor Mustafa ORAL, for his understanding and assistance, as well as all the valuable lecturers of the Department of Computer Engineering, Çukurova University.