

# **Volunteer Finder**

Supervisor: Viktor Kulikov

Students: Feras Kanaan , Muhammad  
Awawdih

## Contents

<b>Project Overview</b> .....	4
Abstract .....	4
Background .....	4
Project Goal.....	4
Key values .....	4
App diagram .....	5
Description:.....	6
approaches .....	8
User authentication.....	8
Signup .....	8
Sign in .....	8
Maps .....	8
libraries.....	9
Manifest and permissions .....	9
Data structures.....	10
Firestore .....	10
for users:- .....	10
for activities:- .....	10
for messages(notifications): - .....	11
Firebase storage .....	13
Activities.....	15
Main activity.....	15
Handle google sign in activity .....	16
Sign up activity.....	17
Log in activity.....	18
User menu activity .....	19
Profile fragment .....	20
Create activity fragment .....	24
Join activity fragment.....	27
Inbox fragment.....	29
Maps activity .....	31
Show on map activity .....	32
Layouts.....	33
Activity layout (activity_one_row).....	33
Profile layout .....	34

Message layout.....	34
---------------------	----

# Project Overview

## Abstract

Mobile application for searching and managing volunteers, setting and managing tasks for them.

## Background

A lot of people have the desire to volunteer, but it not always easy to find the right activity due to their schedule, effort, distance and other factors.

## Project Goal

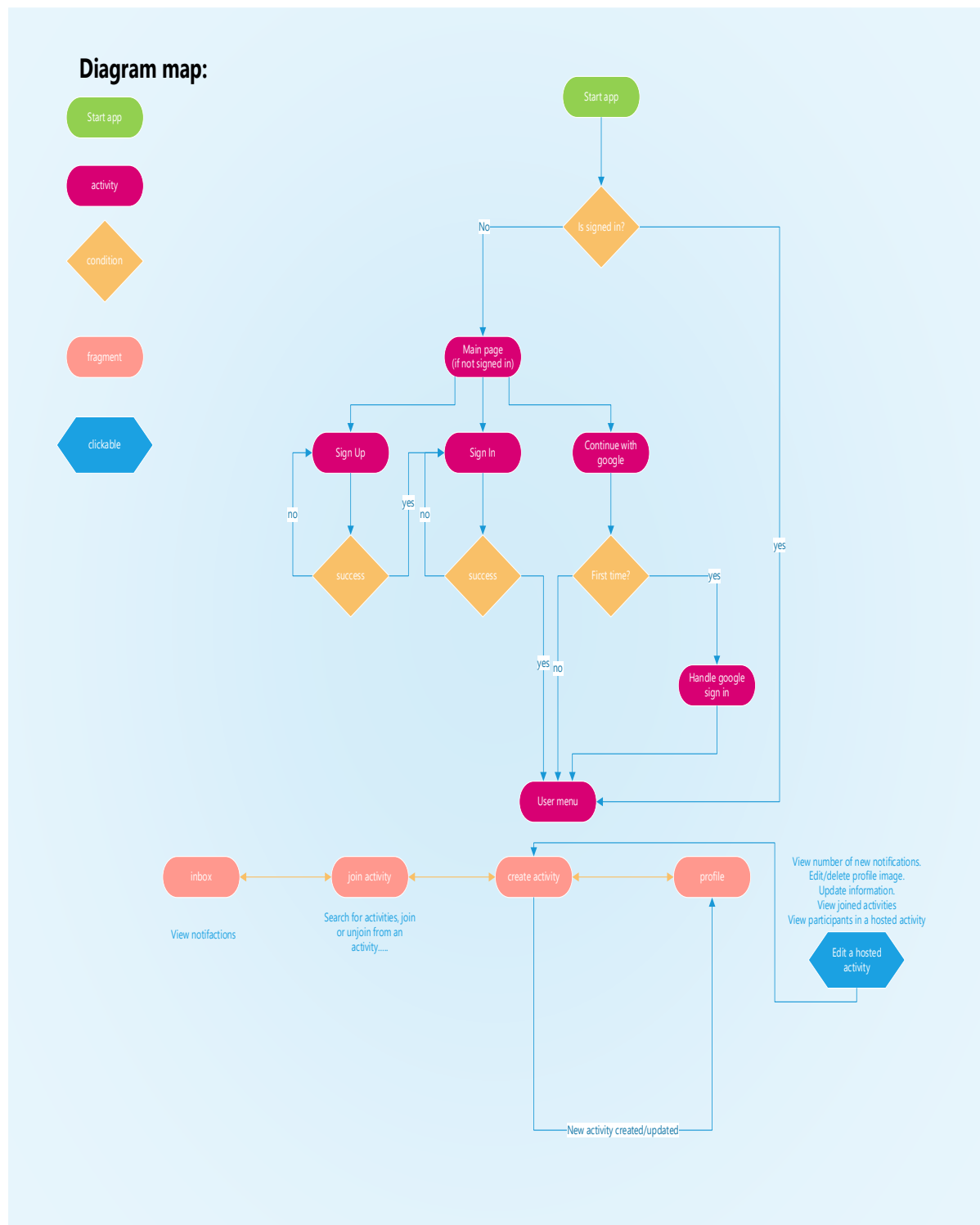
Our goal is to build an android application which will make it easier to find volunteers for a specific activity and to encourage and make it easier for volunteers to participate in a volunteering activity.

## Key values

- Make it easy to find a specific volunteering activity
- Timeserving (use your free time with something)
- Help people and do good to the community

- Make it easier to find volunteers for your activity
- Encourage people to volunteer in their time

## App diagram



## Description:

- The application support user **signs up** with email and password, or using existing google account.
- The user has the ability to sign in and out from their account which enables **multiple users** use the same device.
- Each user has a **profile page** (default page) which contains:
  1. The number of new notifications in inbox.
  2. View and edit/delete current profile image.
  3. Update user information: gender and user skills.
  4. View hosted activities, number of participation in each activity and view participation profile.
  5. View joined activities and cancel enrollment in the activity if needed.
- Each user can search for an activity under the tab of **join activity**:
  1. The user can view the activities in list view (by default) or map view.
  2. The user can join and unjoin from a specific activity.
  3. The user can sort the activities by date, name or location.
  4. The user can search for a specific activity after providing a text to search.
  5. The user can filter the shown activities according to the information he provided: skills, gender, age.
  6. The user can filter the shown activities by their expected effort and time.

- Each user can host a new volunteering activity under the tab of **create activity**, by filing:
  1. Name of the activity.
  2. Description of the activity.
  3. Expected effort and time.
  4. Preferred age for participants.
  5. Category, the categories match the user skills.
  6. Starting date and time.
  7. Location if its relevant otherwise check the option of *no presence required*.
  8. gender segregation(optional).
  9. image for the activity(optional).

- Each user can view the in-app notification\_he got under **inbox** tab:

The user gets a new notification if:

1. An activity he joined was deleted and view the deleted activity.
2. An activity he joined was edited and view the old and new activity information.
3. Another user has joined or Un-joined a hosted activity and view the user profile and the activity.

# approaches

## User authentication

### Signup

With the help of firebase authenticator, we used two methods for registering a new user: -

- 1- Sign up using email and password.
- 2- Sign up with google Sign-In.

### Sign in

With the help of firebase authenticator, we check if there is a current user signed in

(google account or regular account) in this case we redirect the user to his profile page , otherwise the user will be redirected to the application main page.

And he can sign in:

- 1- Sign in using email and password.
- 2- Sign in with google Sign-In.

## Maps

We used the google maps API to create the google maps instances that are needed.

First we generated a new API key for google maps, a billing google account is required!:

<https://developers.google.com/maps/documentation/android-sdk/get-api-key>

following this guide, we added the API to our application:

<https://developers.google.com/maps/documentation/android-sdk/start>



## libraries

- Firebase libraries: common, [auth](#), [firestore](#) and [storage](#).
- Android libraries: core, app compat, material, constraint layout, lifecycle and navigation to enable standard layout and in app actions manipulations.
- Glide, espresso and Picasso to compress, upload and load images from and into the application.
- Android mail : decided not to use since we can't use the Gmail domain starting on 2022.
- Prettytime: to manipulate and customize date and time pickers.
- Google services: [maps](#) described above, and [auth](#) to authenticate google users sign in.

## Manifest and permissions

We enabled three main permissions in our app:

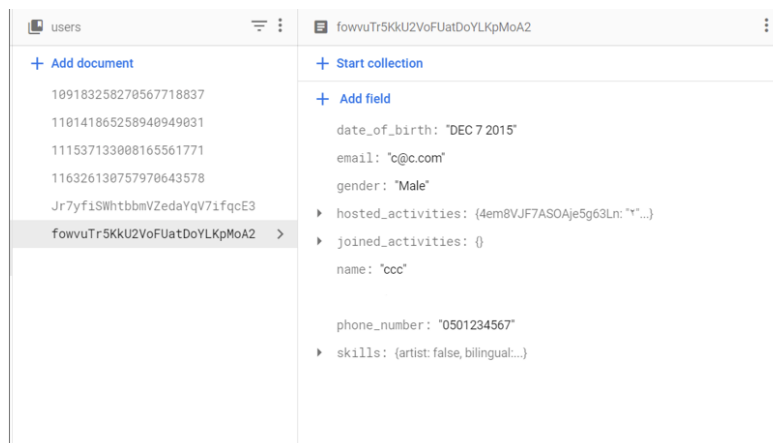
- android.permission.INTERNET : to connect to Internet.
- ACCESS\_COARSE\_LOCATION and ACCESS\_FINE\_LOCATION: to enable access for current location.

# Data structures

## Firestore

We used Firebase Firestore to store our 3 main data structure: -  
for users:-

we saved every user information in the users collection, with the ID given by Firebase authenticator at sign up or google Uid for google accounts.



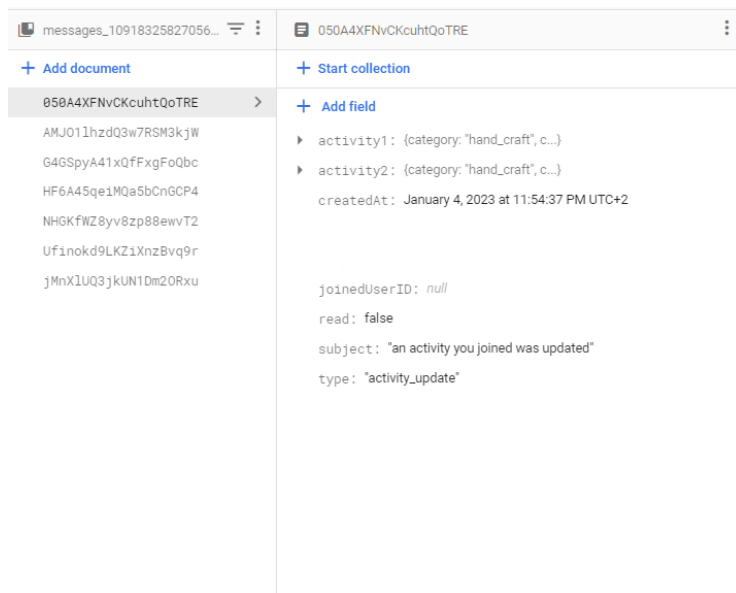
As we see in the photo, under every user ID we saved the user information:

- name, email and phone number for contacting the user.
- Gender, skills and date of birth (age): for filtering the matching Volunteering activities.
- Hosted and joined activities: to bind the users and activities databases.

for activities:-

we saved every activity information in the activities collection, with automatic ID.

we saved each user messages in a collection named `Messages` `userId` .

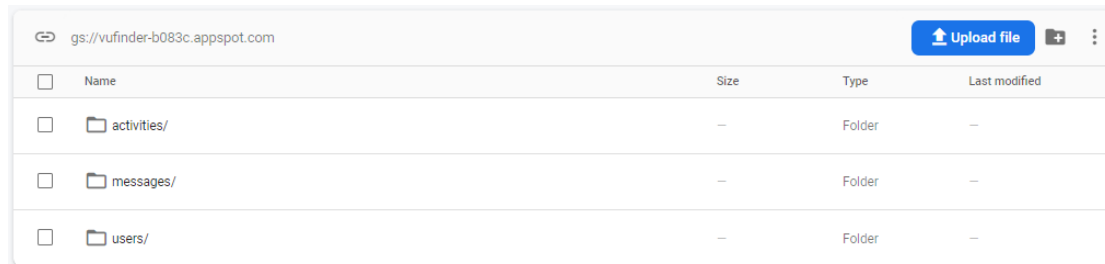


As we see in the photo, under every messages\_user Id we save the information:




- Type and subject to define the notification.
- when there is a notification about an activity, we save a frozen image of that activity as it might change in the future.
- when there is an update notification about an activity, we also save a frozen image of that new activity.
- Created at (time stamp) for sorting the notification.
- Read for coloring the messages that has been read by the user.

## Firestore storage

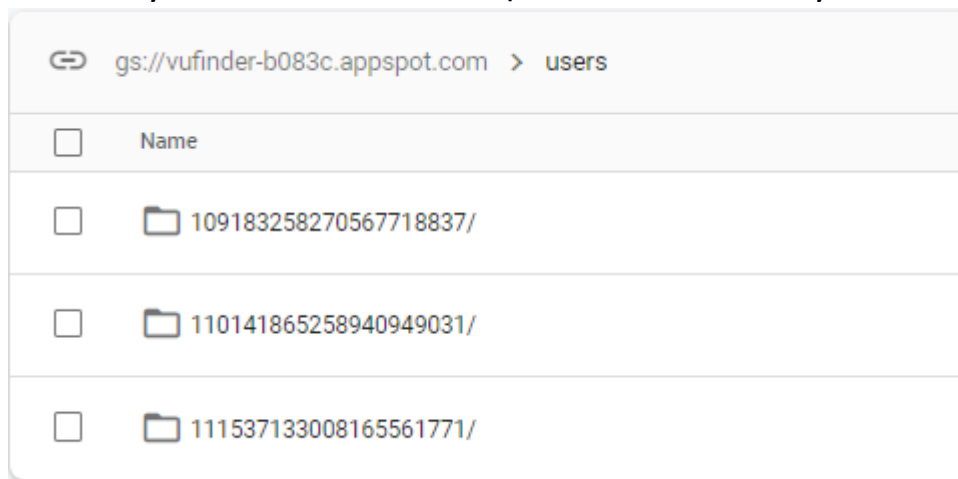
To save images for activities, users and messages (notifications) we used the Firestore storage:






The screenshot shows the Google Cloud Storage interface for the bucket `gs://vufinder-b083c.appspot.com`. It features a table with columns for Name, Size, Type, and Last modified. There are three folders listed: `activities/`, `messages/`, and `users/`. An 'Upload file' button is visible in the top right corner.

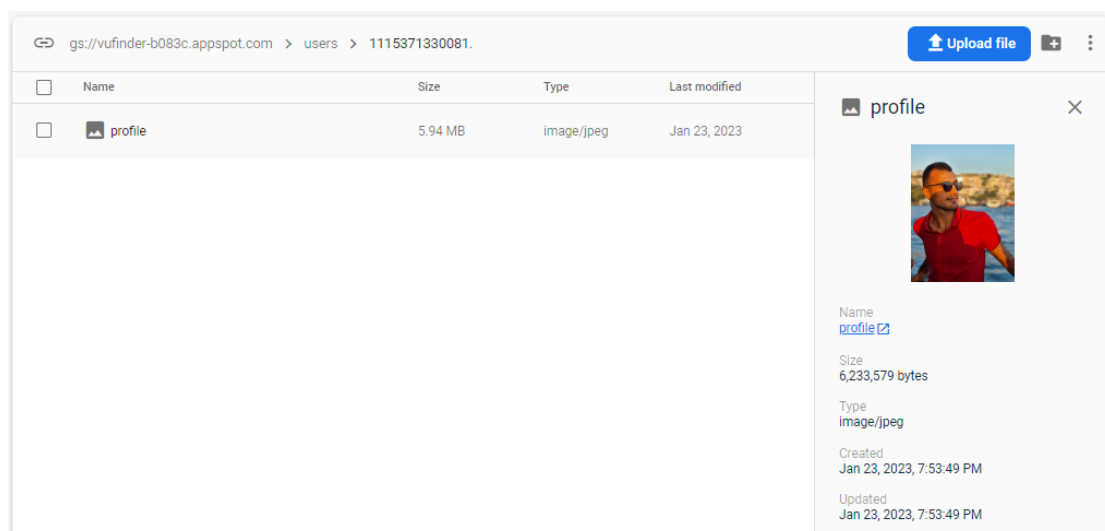
<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 <code>activities/</code>	—	Folder	—
<input type="checkbox"/>	 <code>messages/</code>	—	Folder	—
<input type="checkbox"/>	 <code>users/</code>	—	Folder	—

Each activity and user image (only one) are saved inside a directory named with the id (for user or activity accordingly):




The screenshot shows the Google Cloud Storage interface for the directory `gs://vufinder-b083c.appspot.com > users`. It displays a table with columns for Name, Size, Type, and Last modified. Three folders are listed, each named with a long alphanumeric ID: `109183258270567718837/`, `110141865258940949031/`, and `111537133008165561771/`.

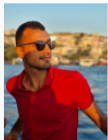
<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 <code>109183258270567718837/</code>			
<input type="checkbox"/>	 <code>110141865258940949031/</code>			
<input type="checkbox"/>	 <code>111537133008165561771/</code>			



The screenshot shows the Google Cloud Storage interface for the directory `gs://vufinder-b083c.appspot.com > users > 1115371330081`. It displays a table with columns for Name, Size, Type, and Last modified. A single file named `profile` is listed with a size of 5.94 MB and type `image/jpeg`. On the right side, a detailed view of the `profile` file is shown, including a thumbnail image and metadata such as Name, Size (6,233,579 bytes), Type (image/jpeg), Created date (Jan 23, 2023, 7:53:49 PM), and Updated date (Jan 23, 2023, 7:53:49 PM).

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 <code>profile</code>	5.94 MB	image/jpeg	Jan 23, 2023

**profile**



Name  
[profile](#)

Size  
6,233,579 bytes

Type  
image/jpeg

Created  
Jan 23, 2023, 7:53:49 PM

Updated  
Jan 23, 2023, 7:53:49 PM

The messages directory is very similar, each image is saved inside a directory named with message id. The images will be named `image1` and `image2` inside this directory (up to two

images can be added). The images of the notifications are compressed to preserve fast performance and save storage.

The shown images on app are not downloaded and loaded to the application via download URL. The pictures are loaded with the help of the glide library.

# Activities

## Main activity



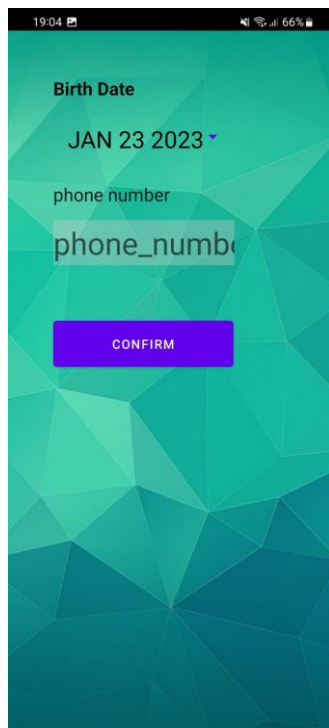
The application main activity, this activity is started when:

1. The user starts the application, and he is not signed in on the device.
2. The user signs out using the sign out button on menu (in [user menu activity](#)).
3. Home button on sign up activity.
4. Home button on log in activity.

The activity contains three buttons, first two are [sign up](#) and [log in](#) and used for navigating to the required activity. The third button is continue with google account and it starts google Sign-In activity (built in), this button is used for:

1. Sign up a new google account if this account UId is not in the Firebase Firestore users database. After the account is authenticated with the help of Firebase authenticator, [handle google sign in activity](#) will start.
2. Sign in with already existing (in database) google account and start [user menu activity](#).

## Handle google sign in activity

A screenshot of a mobile application interface with a teal geometric background. At the top, the status bar shows the time 19:04, signal strength, and 66% battery. The form contains a 'Birth Date' label, a date picker showing 'JAN 23 2023', a 'phone number' label, a text input field containing 'phone\_number', and a purple 'CONFIRM' button at the bottom.

The user (google account) is asked to fill in his date of birth and phone number, since this information is needed for the application, and it's not provided by google.

When the button confirm is clicked:

Check if the provided number is valid: The phone number should be 10 digits number starting with 05 prefix. And that the user is at least 5 years old.

navigate to [user menu activity](#) on Success.



## Sign up activity

The image displays two screenshots of a mobile application's sign-up screen. The left screenshot shows the initial state with a home button at the top left, a 'LOG\_IN' button, and input fields for 'Full Name', 'Email', 'phone number', 'password', 'confirm password', and 'Birth Date'. The right screenshot shows the same form with a 'SIGN\_UP' button at the bottom and a date picker set to 'JAN 23 2023'.

The activity contains two navigation buttons:

1. Home button to navigate to [Main activity](#).
2. Log\_in button to navigate to [Log in activity](#).

When the sign-up button is clicked, check if the information provided is valid:

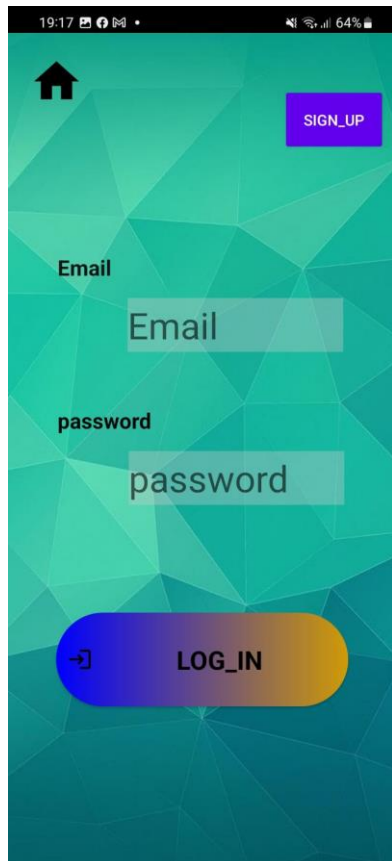
1. Password should be at least 8 characters, contain at least one lowercase and one uppercase letter and contain at least one special character and one or more numbers.
2. Password and confirm password fields are identical.
3. The phone number should be 10 digits number starting with 05 prefix.
4. The user is at least 5 years old.

On success:

1. Add the user provided information to the [users database](#).
2. Navigate to [log in activity](#) and pass the email via intent (user is only asked to fill in his password).

On failure show proper error message using toast.

## Log in activity



If the intent when reaching the activity is not empty this means we got from sign up activity and the information passed is the Email which will be automatically filled.

The activity contains two navigation buttons:

1. Home button to navigate to [Main activity](#).
2. Sign-up button to navigate to [sign up activity](#).

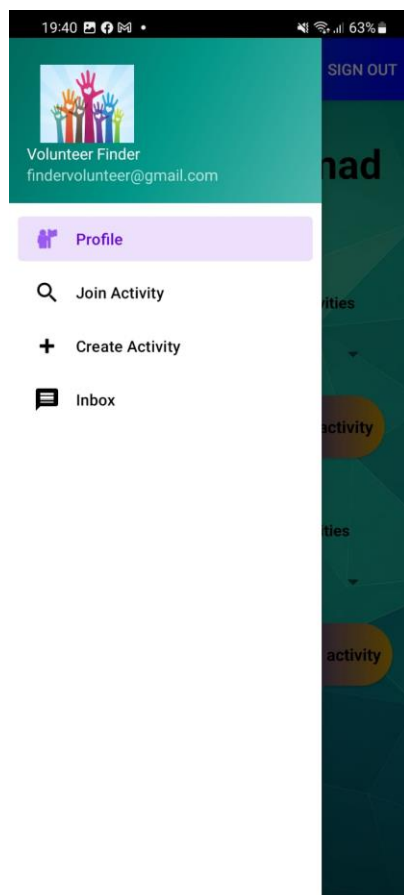
When the log in button is clicked, check if the information provided is not empty and try to authenticate the user with Firebase authenticator.

On success:

Navigate to [user menu activity](#).

On failure show proper error message using toast.

## User menu activity



By default the [profile fragment](#) is loaded on this activity.

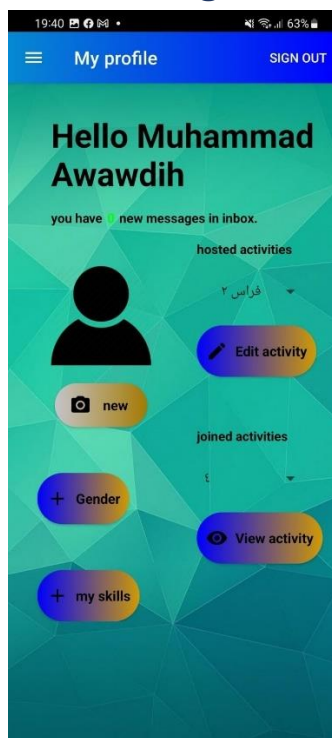
All the data about the user and activities shown on each fragment (user/activity/messages) is loaded from the firebase firestore [databases](#) except for the images which are loaded from the [firebase storage](#).

The menu contains a sign out button:

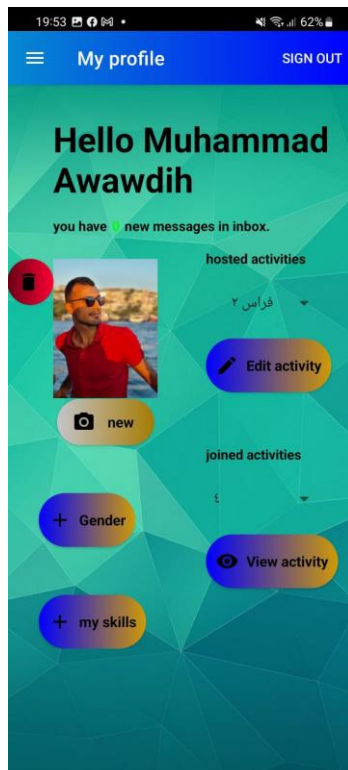
When clicked the user authentication information is deleted (signed out from device) and the [main activity](#) is started.

the navigation menu contains four buttons to navigate between each of these four fragments:

## Profile fragment

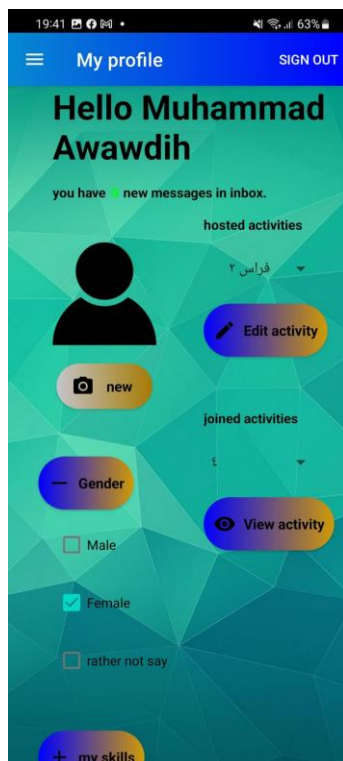


The user can upload a new image from gallery using the new (image) button, the image will be uploaded to [firebase storage](#).



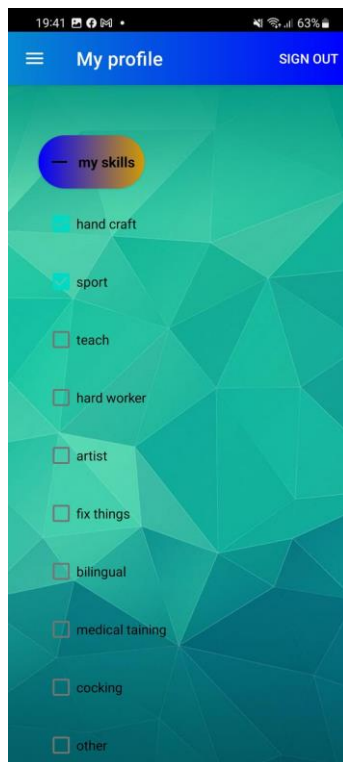
The user can delete his current profile picture by clicking the recycle red button on the top left corner of the image.

When the Gender+ button is clicked, it's replaced with Gender- button and gender checkboxes are shown.



When Gender- is clicked the checkboxes disappear and Gender+ will replace it.

When the skills+ button is clicked, it's replaced with skills- button and skills checkboxes are shown.

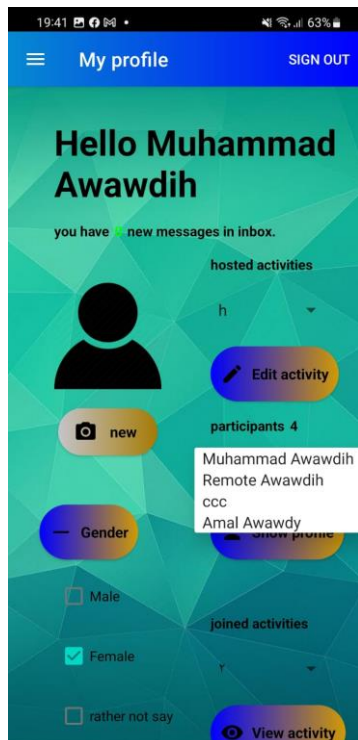


When skills- is clicked the checkboxes disappear and skills+ will replace it.

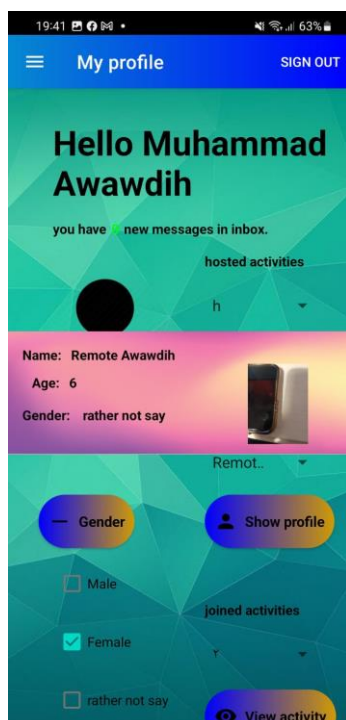
- The chosen skills and gender will automatically be updated in the [users data base](#) (for the specified user) and will be loaded from it the next time he visits the page.

Hosted activities spinner shows all activities ever hosted by the user. The user can choose any activity and edit it using edit activity button, which when clicked will start the create activity fragment (on full page – replacing the menu) with passing the required activity Id.

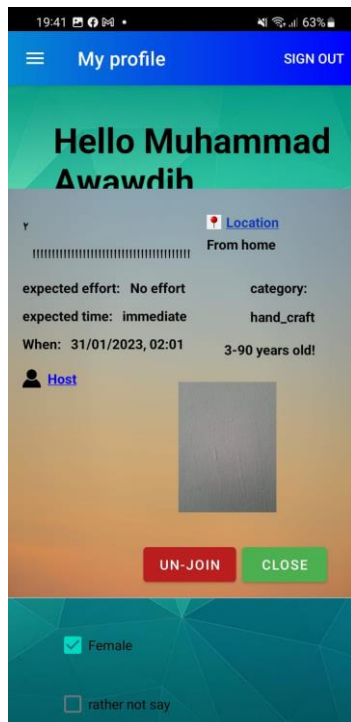
If the selected hosted activity has at least one participant the participants spinner and show profile button are shown:



When clicking the show profile button, the selected participant profile is shown as a pop view, inflating the [profile layout](#):



When the view activity button is clicked the selected joined activity is shown as a pop view inflating the [activity one row layout](#):



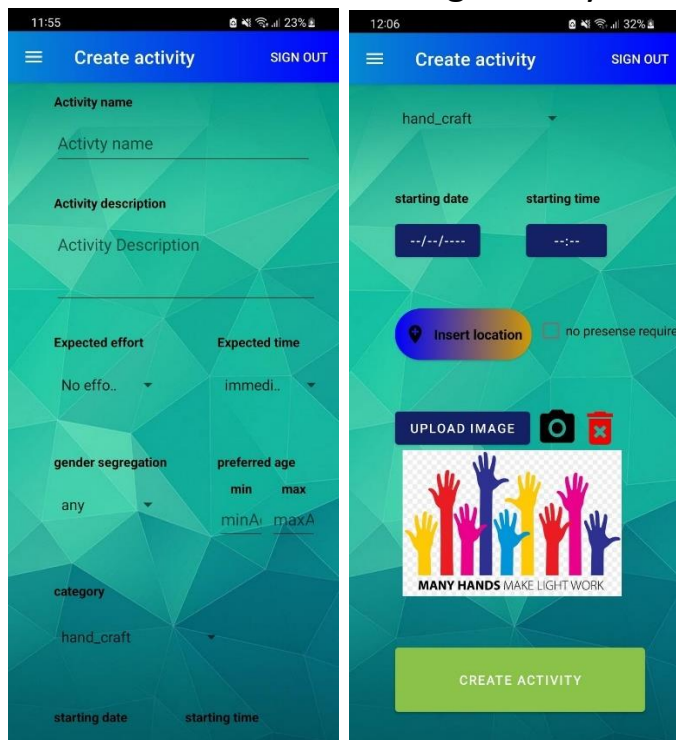
the joined activities list for the user is cleaned every time the profile page is reached: if the joined activity does not exist anymore or the starting date is passed it will be deleted (no longer relevant for this user).

### Create activity fragment

This activity is responsible for two actions:



## 1. Create a new volunteering activity:



the user is asked to fill the information about the volunteering activity he wants to host.

there are two picker dialogs one for starting date and another for starting time. The spinners are used to categorize the activity.

the user can upload a new image from gallery using the upload image button or alternatively upload a new picture from the camera. This uploaded image could be deleted using the recycle button on the top right corner of the image.

the insert location button starts the activity [Maps activity](#) for result. This activity will return the picked location latitude, longitude, country and city.

when the create activity button is clicked, we check if the information provided is valid:

- activity name is not empty.
- activity description is at least 20 letters.
- minimum age is smaller than maximum age, both are positive numbers and less than 100.

- spinners have selected items (should never happen).
- activity starting date and time are not empty and are not from the past.

If the information is not valid a proper message will be shown using the snack-bar.

Else the new activity is added to the [activities database](#) and the user will be redirected to his [profile page](#).

## 2. Update or delete existing volunteering activity:

The image shows two screenshots of a mobile application interface for updating a volunteering activity. The left screenshot displays a form with the following fields: 'Activity name' (with the value 'فراس'), 'Activity description' (with the value 'أبنتيتو بو بنيزو يوزو يتيو يوزو يزي يزي'), 'Expected effort' (with the value 'No effo..'), 'Expected time' (with the value 'immedi..'), 'gender segregation' (with the value 'any'), 'preferred age' (with 'min' 3 and 'max' 39), 'category' (with the value 'hand\_craft'), and 'location' (with a location pin icon). The right screenshot shows the same form with the following values: 'category' 'hand\_craft', 'starting date' '31/01/2023', 'starting time' '09:34', 'location' 'Insert location', and 'no presense require' checked. Both screenshots have 'Cancel' and 'remove activity' buttons at the top. The bottom of the right screenshot features a green 'UPDATE ACTIVITY' button.

the data shown is loaded from the [activities database](#); the meant activity id is passed from the profile fragment using intent.

the create activity button will be changed to update activity and it will update the specified activity information in the database and in addition an update notification will be uploaded to the [messages database](#) of each user who is currently enrolled in this activity.

two more buttons are added to the page, cancel button which will redirect the user to the profile fragment. And the remove activity button which will: delete the activity image

from [Firebase storage](#), delete the activity from the [activity database](#) and send each of the enrolled users in the activity a [notification](#) about the deleted activity.

## Join activity fragment

All (future) volunteering activities are shown in a list view by default:



The page is split into three sections:

1. Footer for pages navigation: each page loads only four (could be changed easily; a class parameter) activities to maintain application performance. The navigation between pages will not effect the current set filters or sorting.
2. Middle section: show the current required activities each inside [activity layout](#).
3. Upper section (collapsible) contains:
  - Google maps button(map view): when clicked a google maps instance will replace the middle section

(activities list view) and all activities containing location and matching the current filters are added to the map as markers. The inflated layout containing the map is the same layout for the [show on map activity](#).

- Three sorting methods; by name, by starting date and time or by distance from current location (if accessible).
- Filters which are described below:



spinners to choose to show only activities that match chosen expected time and effort by the user. keyword search: show all activities containing the text in its name or description. checkboxes to filter the activities to those who match the user provided information: skills, age and gender. The gender checkbox will only be shown if the user didn't choose the Gender: rather not say option on his [profile page](#).

## Inbox fragment



the notifications are loaded from the current user [messages database](#). The unread messages are colored, and the read ones are grayed out.

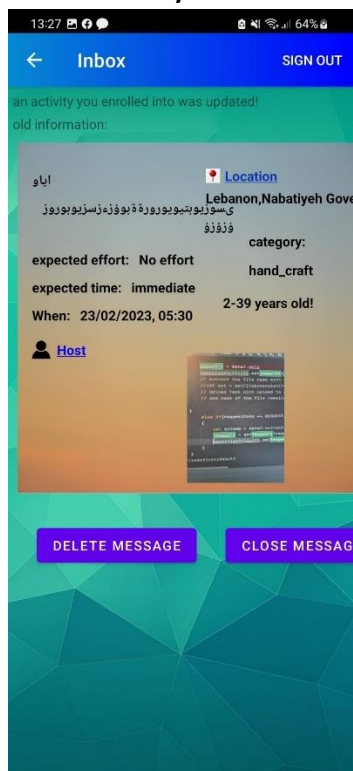
When a message (button) is clicked the layout will be replaced with a [message layout](#) which will be filled with this notification information, there are three types of notifications:

## 1. An activity was updated notification:

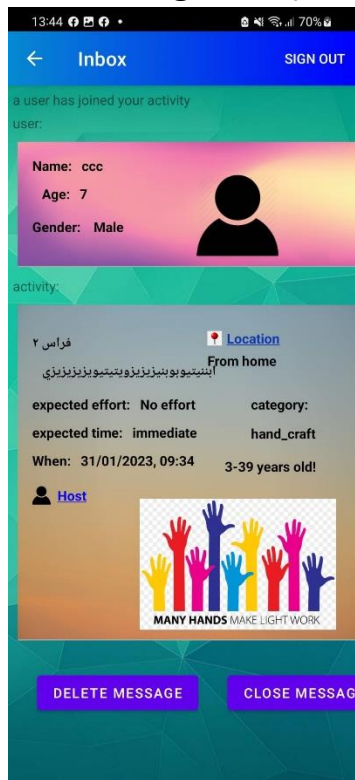


as we can see the location for this activity was updated.

## 2. An activity was deleted:



3. A user has joined or un-joined an activity you are hosting.  
If there are more than one notification from this kind with the same joining user and same activity only the latest one will be kept, and the rest are deleted (from database and storage also).



## Maps activity



A marker will be set on the current location, the user can drag the map to any desired location and click the choose location button which will add the information needed (longitude, latitude, city, and country) to the activity result and end it.

This map and all the maps are loaded from the [google maps API](#).

## Show on map activity



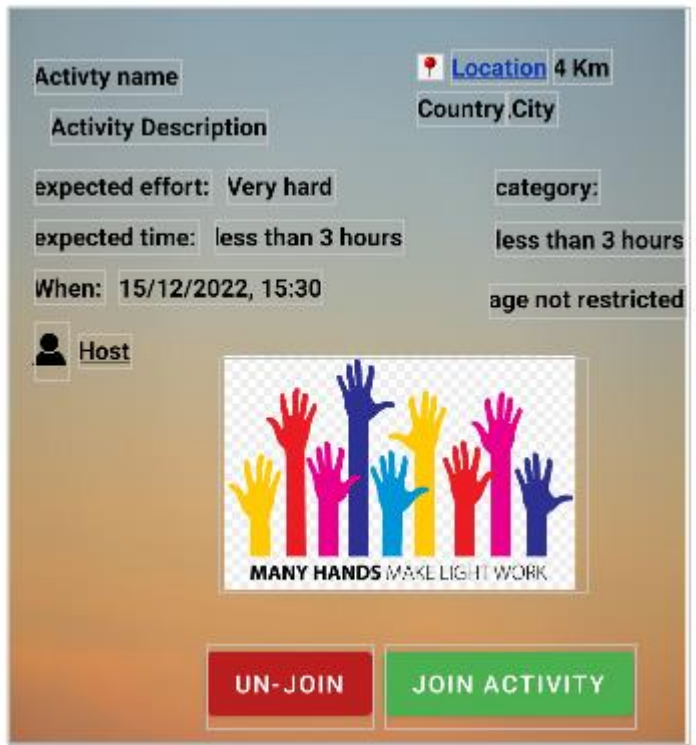
The user can view the location of a volunteering activity, two markers are added one for current location and the other for the activity location (the location is passed via intent).

When the user clicks on the close button the activity will end successfully (and return to the previous activity).



# Layouts

## Activity layout (activity\_one\_row)



The information for the activity to be loaded in this layout is pulled from the [activities database](#) and from [Firebase storage](#).

Clickable texts:

- Host: is only visible when the user who is viewing the activity isn't the host for this activity. When clicked a pop view for the host profile will be shown (inflating [profile layout](#))
- Location: is only clickable and colored when the activity location is specified (otherwise from home text will be shown) and will start the [show on map activity](#). The distance is also only visible when both activity and current locations are accessible.

Buttons:

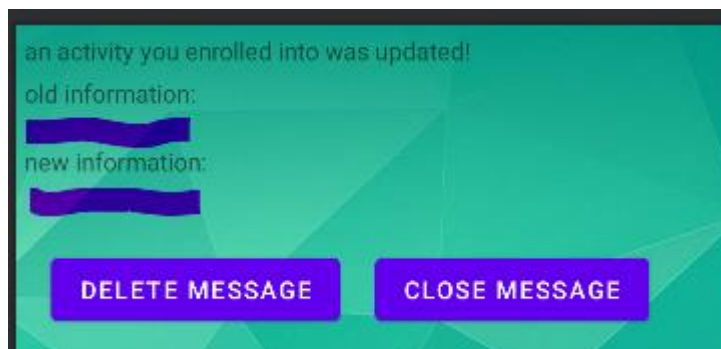
- Un-join: clickable only when the user viewing the activity isn't the host and he is joined for this activity.
- Join activity: clickable only when the user viewing the activity isn't the host and he is not joined for this activity

## Profile layout



The information for the activity to be loaded in this layout is pulled from the [users database](#) and from [Firebase storage](#).

## Message layout



The information for the activity to be loaded in this layout is pulled from the [messages database](#) and from [Firebase storage](#).

The purple markers represent linear layout that can be inflated into with another layout.

The layout will be redesigned according to the message type, and the inflatable layouts will host either a user layout or an activity layout. See [inbox fragment](#) to view the different types of notifications.

There are two buttons in the layout:

- Close message: redirect the user to his [profile page](#).
- Delete message: remove the message from messages database and [messages database](#) and redirect to [profile page](#).