The instruction for *PhotKey* Mobile App

Subject: Computação Móvel

Subject Instructor: Prof. José Maria Amaral Fernandes

Group members (g05)	
Stanisław Franczyk, 112059	Miron Oskroba, 112169

Table of contents

App Introduction

Motivation

Technologies of the 21st century allow us to have a small smartphone that we take with us on a journey, so we remember our memories - with numerous great quality photographies. PhotKey App focuses a group of hobbyists and amateurs of photography in one place. *The Daily Creative Challenge* creates an opportunity to achieve artistic fulfillment of an individual user as well as the desire of gaining the prestige of winning by competing with others.

Daily Creative Challenge

There is a new challenge for users everyday. Users can enroll in a *Daily Creative Challenge* by taking and submitting exactly one photo that best matches the *Daily Category*. Users, who didn't submit their photo to a *Daily Challenge* are not allowed to be a part of the challenge. Using *Swiping Vote Mechanism* users can vote for the *Daily Best Photo*, which would be posted on the official Instagram and reward the *Daily Winner*.

Guidelines

UX is the main guideline along the whole project. The application interface is designed as minimalistic and simple. Users can manage their individual profiles, add friends and view other users' photos.

Features and properties

App features:

- Taking photos - user can take a photo to participate in a daily challenge,
- QR code - each user has its individual QR. Any user can scan it to add to friends,
- Geolocation - with photo upload, the geolocation coordinates are sent, so other users know the location when the photo was taken,
- Shake smartphone - user can shake his smartphone to give photo a star,
- User-friendly UI design - the app was designed and implemented taking the end-user perspective as a main guideline.

Navigation

Navigator version: 1.0 - using Router to route between App states.

1. First thing that is presented to the user is the *Login Page*.
2. The next view is the *Daily Category Page*.
3. From this point, the user is forced to take a photo in order to participate in a Daily Challenge.

4. After a photo is taken the user can see its preview and if he is satisfied with the result, he can decide to tap the ‘Send’ button.
5. The following view is the *Swiping Cards Selection* – return to the *Take a Photo* is impossible. On the return arrow tap, *The Daily Category Page* is presented. From here users can go to *Select Favourites Page*, and after they choose their favourites photos, the *Daily Top Page* is being shown.
6. *User Profile Page* is accessible from pages: *Daily Category Page*, *Swiping Cards Selection, Choose Favourites Page* and *Daily Top Page*. From here, the user can open a *Show QR Page* and *Scan QR Page* to make adding users easy. *User Profile Page* additionally enables the possibility to display a list of friends and view other users profiles.

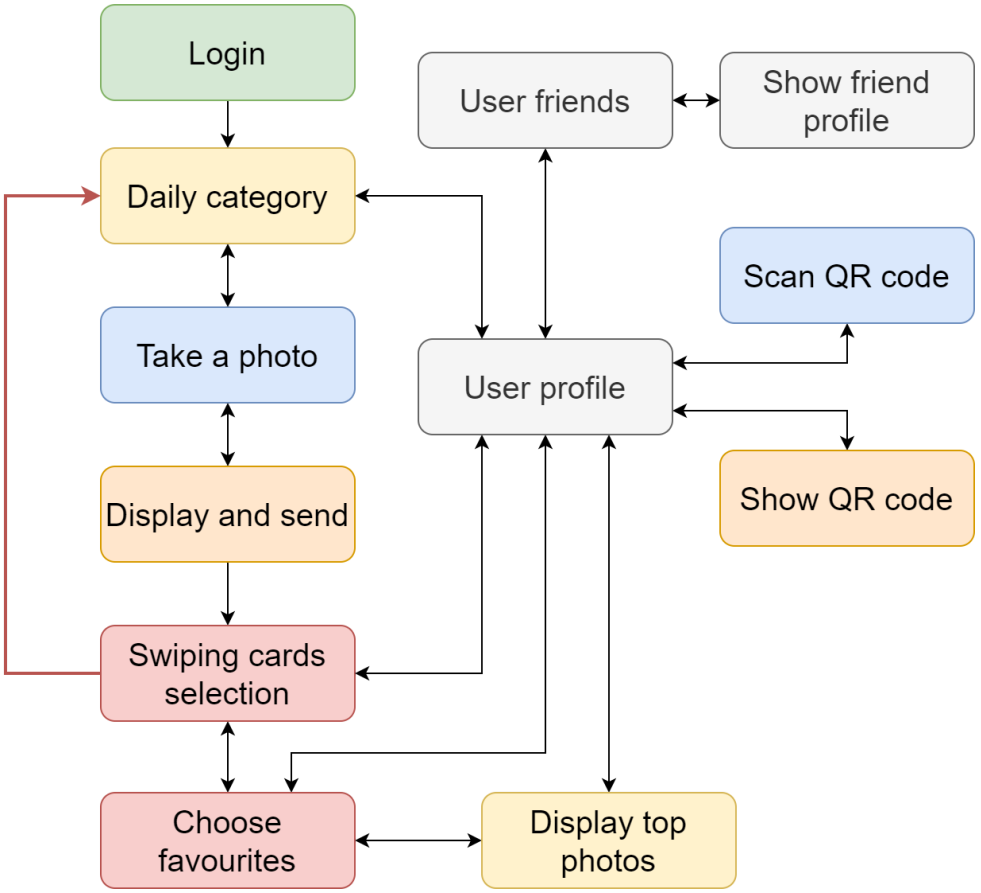


Image 1. Navigation diagram

Architecture Solutions

During the project team members assumed to follow the best OOP practices as well as clean code practices, eg. good naming convention patterns, SOLID, loose coupling, etc.

The architectural decision that facilitates access to application data is a Provider strategy. A good example of its usage are Providers: LocationProvider and CardProvider, although it is an elegant solution to access the data, still some of the objects were hardcoded, which is considered as a very poor practice.

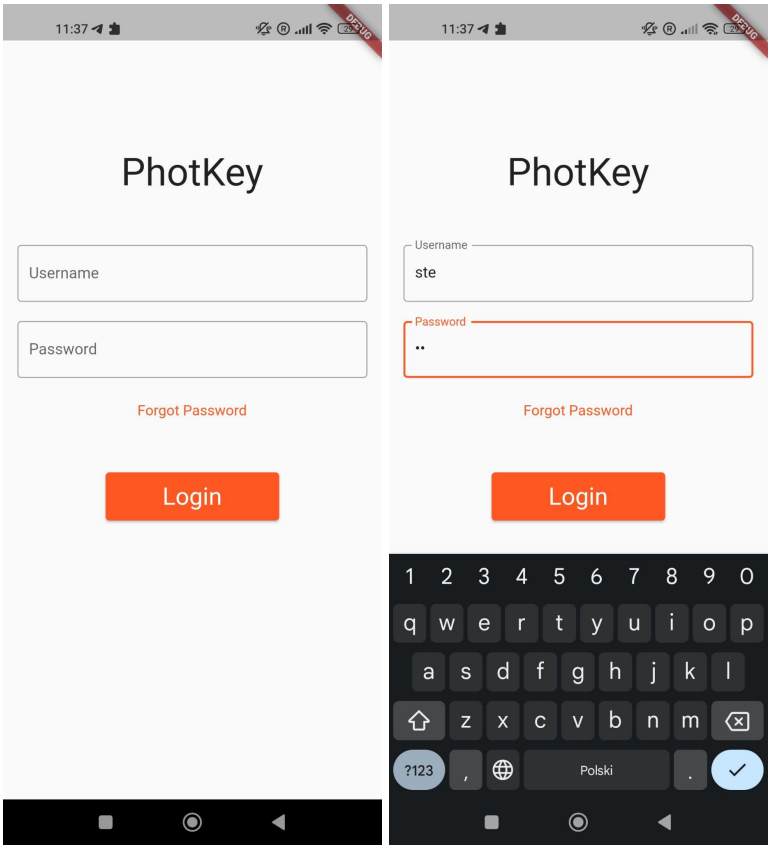
CardProvider stores and shares data of Photos. Thanks to remembering the position of the last PhotoCard it allows the user to do a swipe gesture on SelectingPhotoPage.

LocationProvider enables the widgets to access GPS data. Because it's an asynchronous solution, it needs to be handled using the Future object. To access Future values FutureBuilder Widget was used and handled properly - during awaiting the value, the circular progress bar is being displayed.

User interface

Login page

A standard login page has been created to log into the app.



Images 2. Login Page View

After entering the login and password, and then pressing the button "Login", the application opens the "Daily category" view.

Daily category page

The view is showing the current category. A user's task is to take a photo that will be matched with the displayed category.

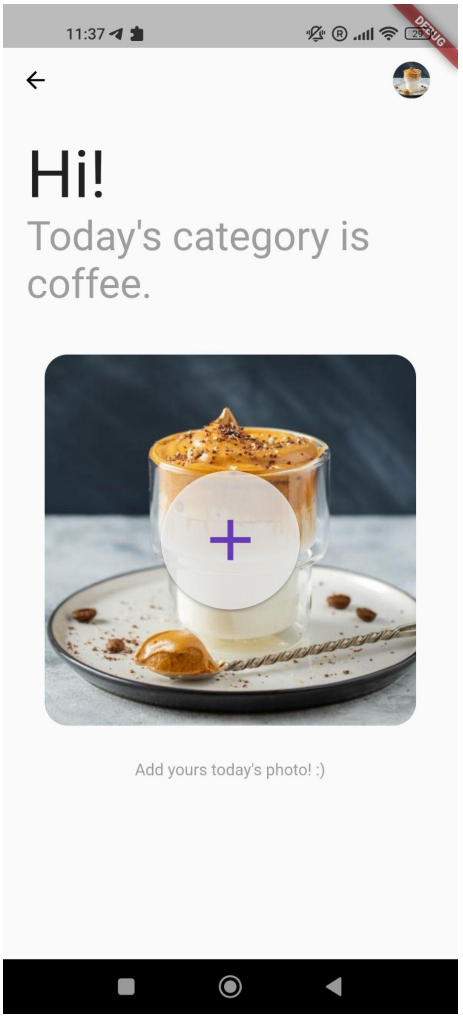
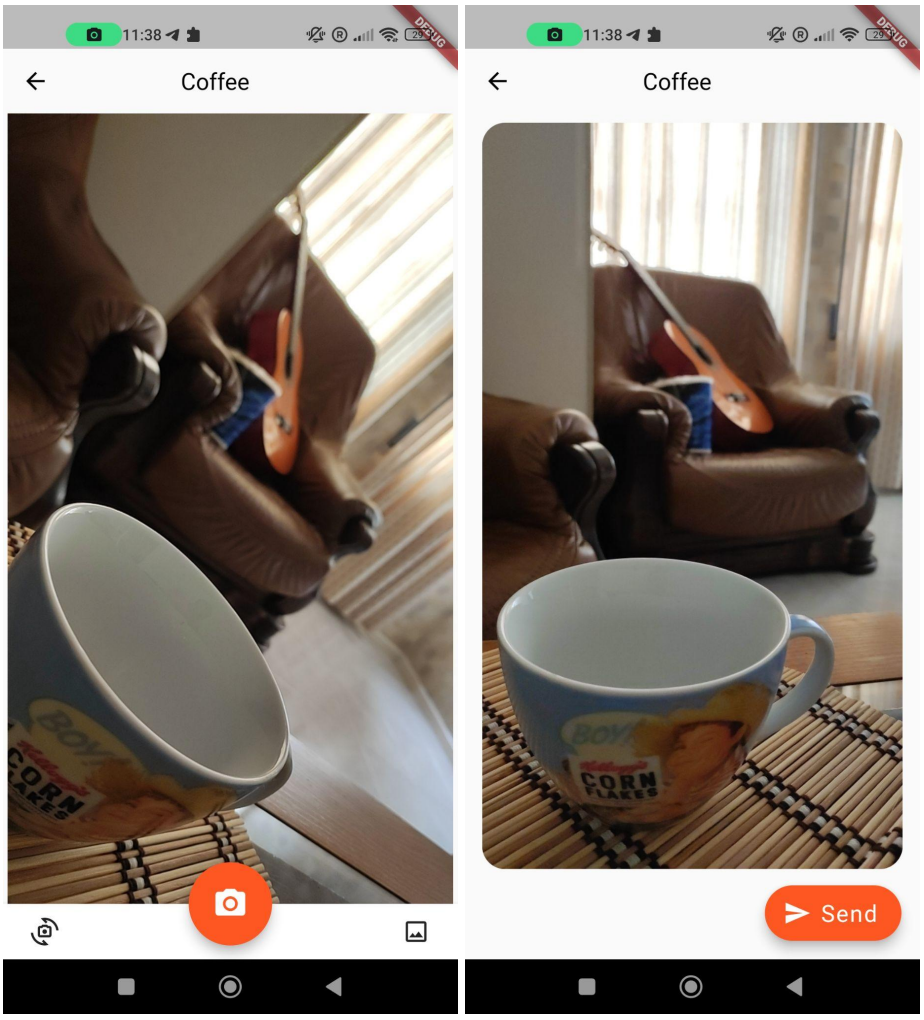


Image 3. Daily Category Page View

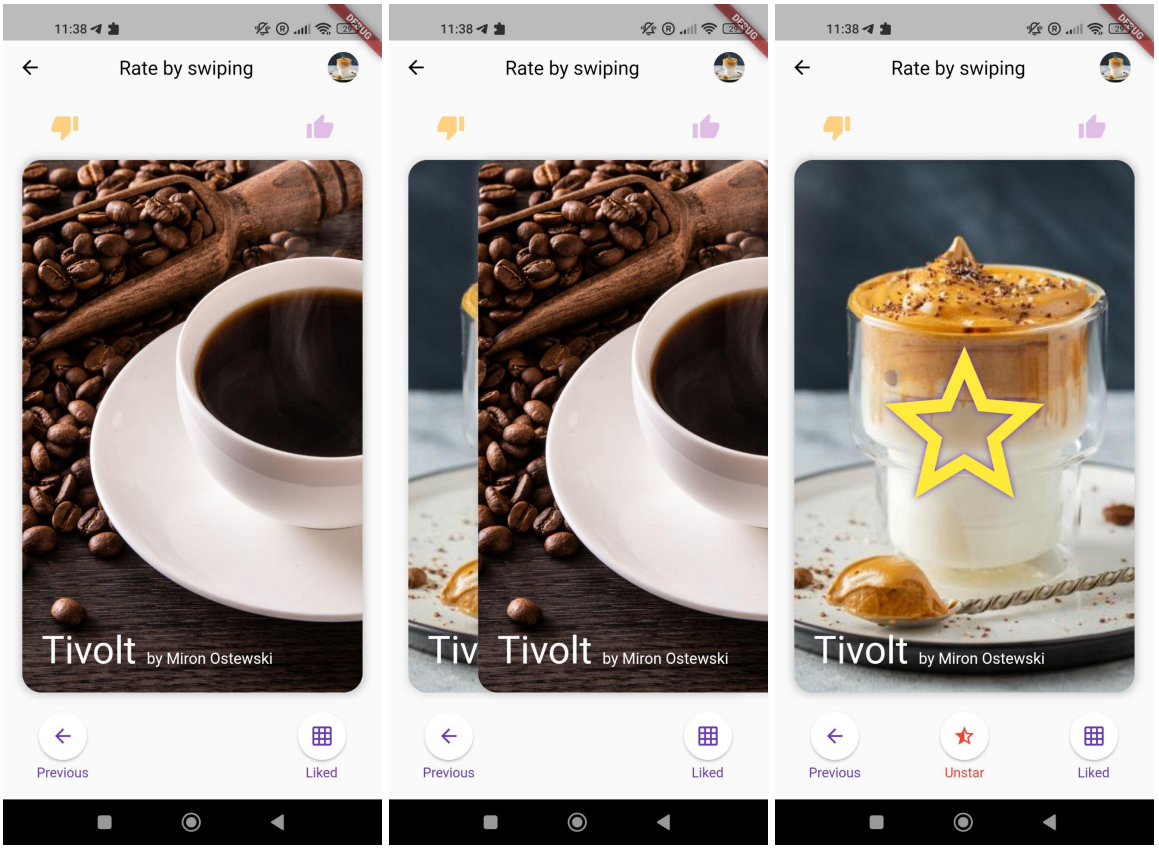
Views for taking a photo and displaying taken photo



Images 4. Pages: Take a Photo, Display Taken Photo Views

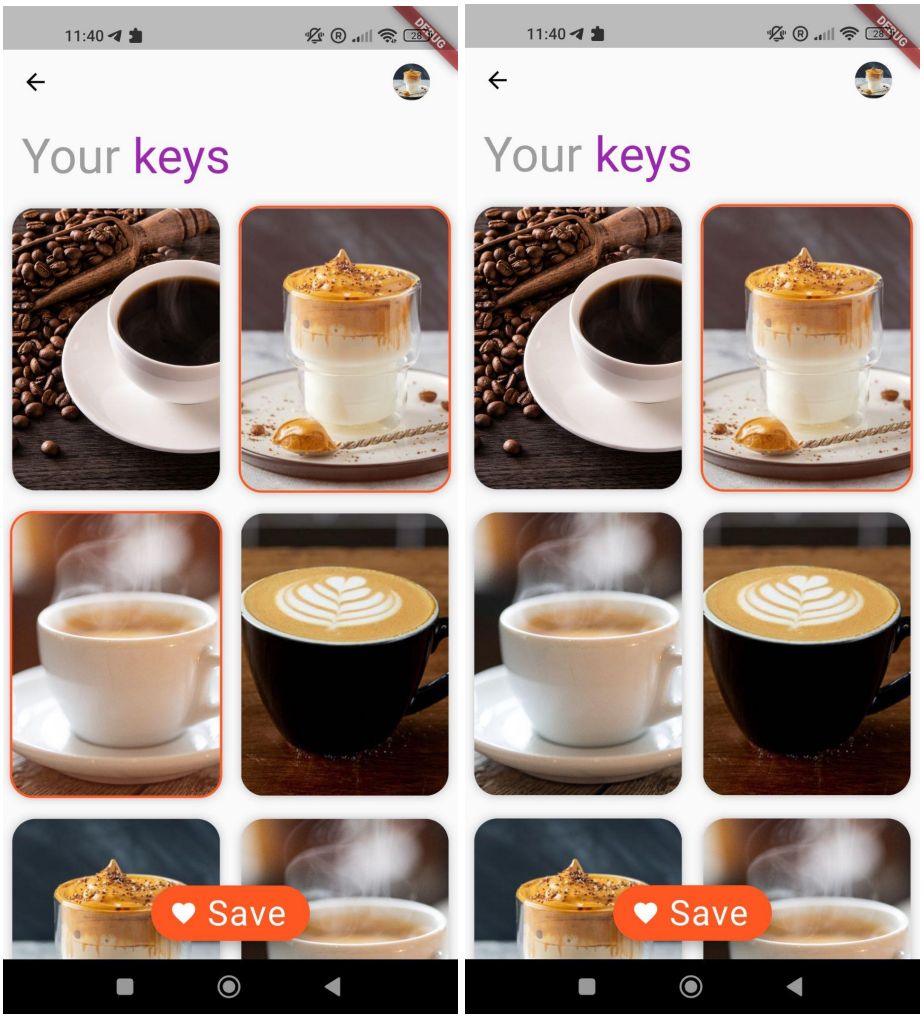
Swiping cards selection

The view of first selection. Here user likes photos that he considered worthy of being the best in the category by swiping to right and rejects others by swiping to left. Liked photos will appear on *Choose favourite photos page*.
User can shake smartphone if he find unusual photos and sign them by a star.



Images 5. Swiping Cards Selection Page View - shake your phone to give stars

Choose favourite photos page



Images 6. Your Keys: left - selected 2, right - selected 1

Daily top page

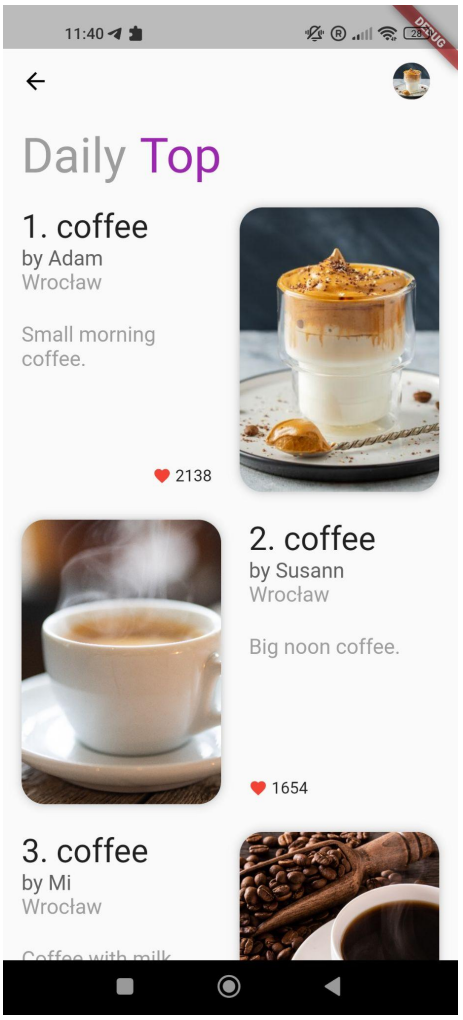
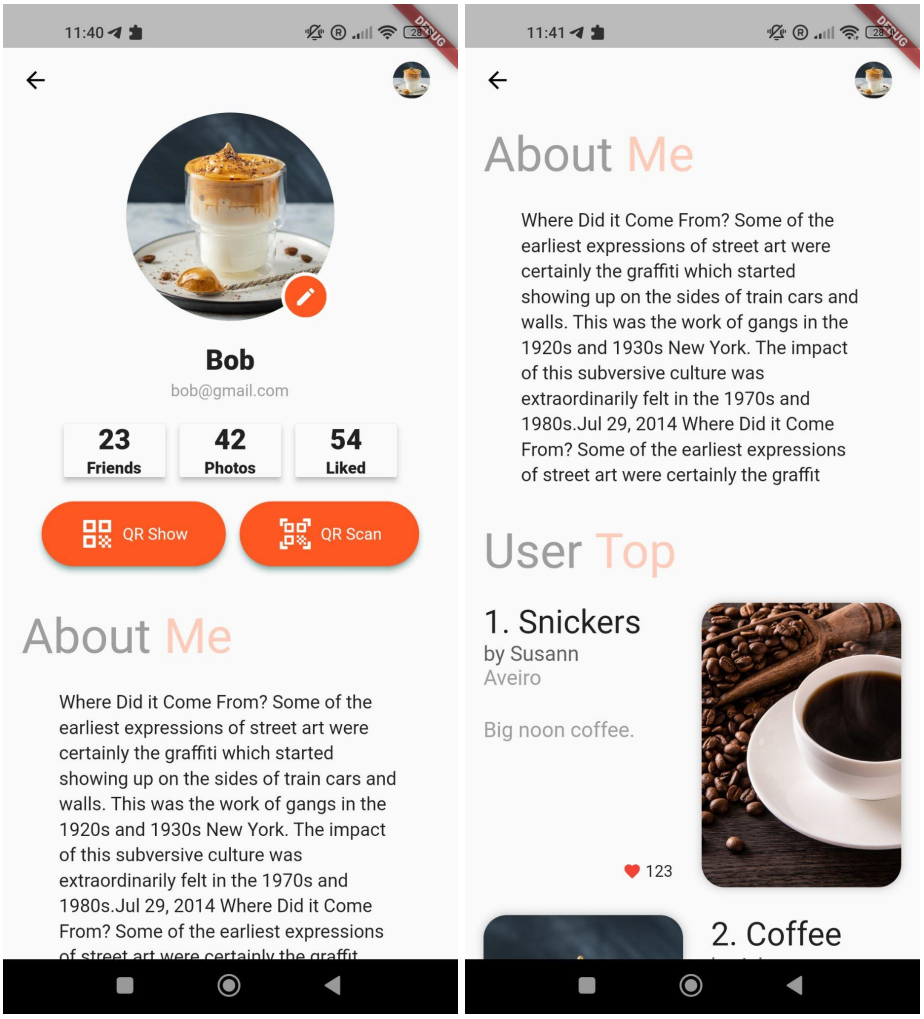


Image 7. Daily Top Page View

User profile page



Images 8. User Profile Page View

Friends page

The view is used to show a list of user friends. After clicking selected friend his profile will be shown.

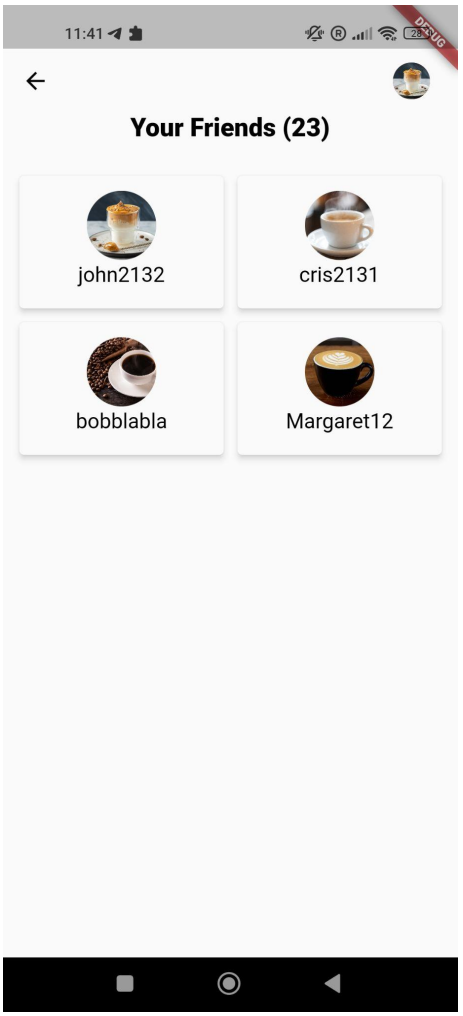
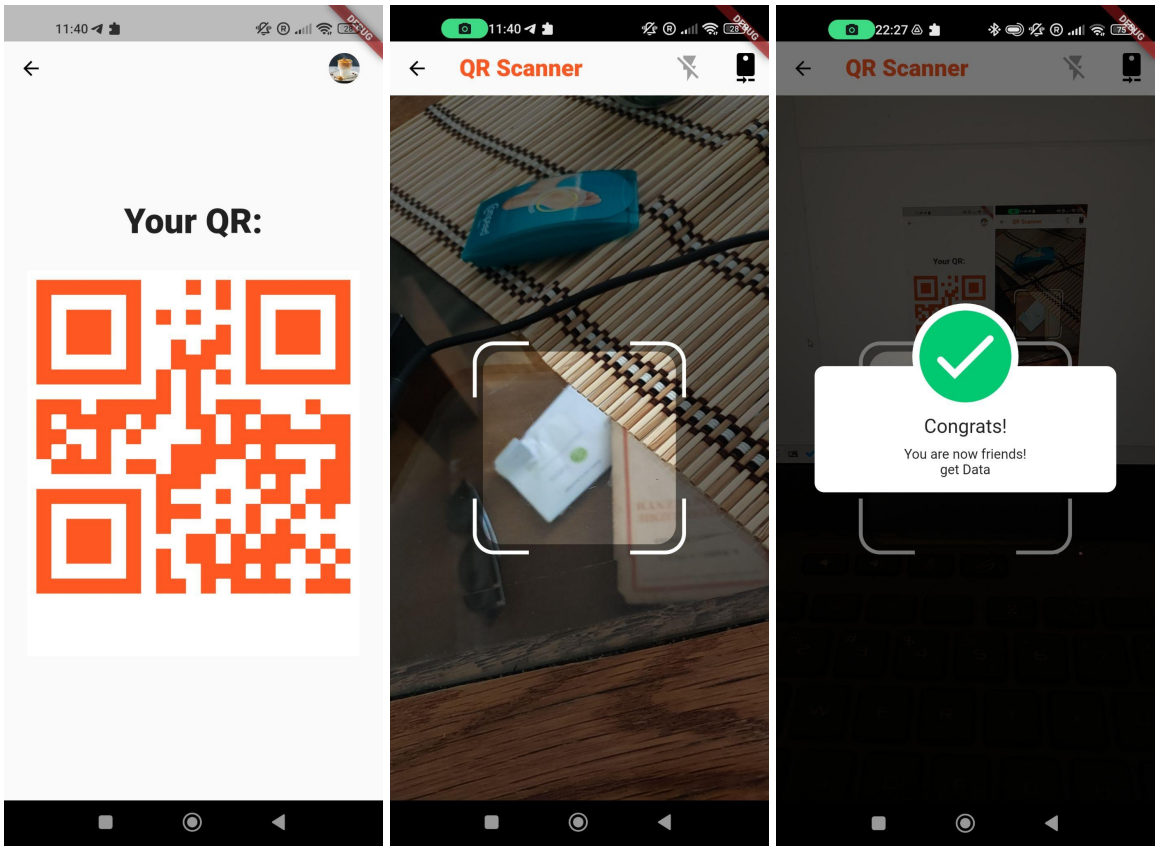


Image 9. Friends Page View

Page show QR code and page take QR code

The pages are used for adding new user friends. User can display their own QR code, which can be scanned by another user of the application.



Images 10. Adding friends using QR code

Project Dependencies

Dependencies used in the project are mandatory in order to run the application. The Github repository already contains the file *pubspec.yaml* which defines all necessary project dependencies and their versions. Used dependencies are as follows:

1. qr_flutter: ^4.0.0 - used for QR frame overlay,
2. mobile_scanner: ^2.0.0 - used to handle QR logic and camera,
3. awesome_dialog: ^3.0.2 - used to alert animated dialogs, UX purposes,
4. cupertino_icons: ^1.0.2 - used for icons,
5. provider: ^6.0.3 - used for providers,
6. http: ^0.13.5 - used to handle http requests,
7. camera: - used to take photos using camera,
8. image_editor: 1.1.0 - fixed wrong displaying from front camera,
9. shake: ^2.1.0 - used to handle mobile shake for super like,
10. flutter_animate: ^2.0.1 - shake animation,
11. geolocator: ^9.0.2 - used to get current mobile location

Project Code

Project's code is located on git repository and is available via public link: https://github.com/sfranczyk/photkey-frontend.

Contribution Assessment

As a group, we agreed that the contribution was equal for each member of the team.
App features:

- Taking photos - Stanisław
- Shake smartphone - Stanisław
- QR code - Miron
- Geolocation - Miron

App Pages:

- Login Page - Stanisław
- User Profile, User Friends - Miron
- User Avatar, User Top - Miron
- Cards Swiping Selection - Stanisław
- Select Favourite Page - Stanisław
- Scan/Show QR - Miron
- Take a Photo Page, Display Photo Page - Stanisław
- Daily Top Page - Stanisław
- Shake and star animation - Stanisław
- Animated Alerts - Miron
- Card Provider - Stanisław
- Location Provider - Miron

Achieved Objectives

Every preassumed technical feature has been implemented and verified, although the API backend service has not been configured with the app. In order to improve the app in the future and make it fully functional, the backend API could be integrated with the Flutter code using eg. retrofit dependency that could facilitate this process. The spring-boot backend code is available via link here.