

Bachelor MKI
„Mobile Computing“,
SoSe 2022
Prof. Dr. Natividad Martínez Madrid

- Project Documentation -

UkrAid – Matching Offers and Needs

Submitted by:
Emanuel Petrinovic (767126)
Lisa Tochtermann (767267)
6. Semester

Contact address:

Emanuel.Petrinovic@Student.Reutlingen-University.DE

Lisa.Tochtermann@Student.Reutlingen-University.DE

Submitted on: 29.06.2022

Abstract: Goal of this project is to design a mobile application that helps refugees from Ukraine to get things they need from resident donors. The application contains a user management as well as the possibility to make advertisements as well for people who need help and for the donors. To overcome language barriers, an automatic translation is implemented. The application adapts to the language the user has set up in his phone. The document describes the process of development with Flutter for Android devices.

Table of contents

1	Introduction	3
	<i>This chapter covers the idea and motivation as well as the goals of our project.</i>	3
1.1	<i>Idea and Motivation</i>	3
1.1	<i>Goals</i>	3
2	Requirements Engineering	4
2.1	<i>Storyboard</i>	4
2.2	<i>Personas</i>	4
2.2.1	Persona 1	5
2.2.2	Persona 2	6
2.3	<i>User Stories</i>	7
2.4	<i>Use Cases</i>	9
2.4.1	Search an item	9
2.4.2	Create an advertisement	10
2.4.3	Contact advertiser	11
2.4.4	Manage advertisements	12
2.5	<i>Requirement Specification</i>	13
2.5.1	Functional Requirements	13
2.5.2	Non-Functional Requirements	20
3	Conceptual model	22
3.1	<i>Component diagram</i>	22
3.2	<i>Sequence diagram</i>	23
3.2.1	Create User	23
3.2.2	Create Ad	24
3.3	<i>Mock-Ups</i>	25
3.3.1	Login Screen	25
3.3.2	Sign up Screen	26
3.3.3	Home Screen	27
3.3.4	Detail Advertisement Screen	28
3.3.5	Create Advertisement Type Offer Screen	29
3.3.6	Create Advertisement Type Searches Screen	30
3.3.7	Profile Overview Screen	31
4	Software Architecture	32
4.1	<i>Layered Pattern</i>	32
4.2	<i>MVC Pattern</i>	33
4.3	<i>Repository Pattern</i>	34
4.4	<i>Dependency Injection</i>	36
4.5	<i>Dependencies</i>	37
4.6	<i>Database</i>	38
4.7	<i>LibreTranslate</i>	39
5	Result	40
5.1	<i>Features</i>	40
5.1.1	Signup and Login Process	40
5.1.2	Editing the User Profile	41
5.1.3	Create Advertisement (offer type)	42
5.1.4	Create Advertisement (searches type)	43
5.1.5	Advertisement Management, Edit and Delete	44
5.1.6	Search Advertisements	45
5.1.7	Home screen filtered by search	46

5.1.8	Example localization	47
5.2	<i>Degree of completion of the requirements</i>	48
5.3	<i>Feedback</i>	49
5.4	<i>Challenges</i>	50
5.5	<i>Conclusion</i>	51
6	Table of Figures	52
7	Table Directory	53
8	Declaration on oath	54

1 Introduction

This chapter covers the idea and motivation as well as the goals of our project.

1.1 Idea and Motivation

Many refugees are entering Europe/Germany because of the war in Ukraine. A lot of those people lack many necessities like hygiene products, clothes, various electrical devices etc. thus relying on donations from other people.

The organisation of those donations is normally handled by organizations like “Rotes Kreuz”, “Caritas”, “Diakonie” etc. But oftentimes these organisations are not trusted by the donors, or the donations never reach the actual people. The goal of our application is to connect donors and recipients directly to each other. With this, the two parties can organize the donations between themselves without having to rely on a 3rd party. The application is basically an advertisement board for people to advertise their donation offers or potential recipients to advertise their needs. This makes it extremely easy for users on either side to make immediate contact with the other side without the help of an organisation. This documentation describes the technical development of the UkrAid app.

1.1 Goals

The goal of this project is to give refugees an application that helps them in the integration process. This app is supposed to give residents and refugees a way to find contact each other, and therefore help each other out.

2 Requirements Engineering

2.1 Storyboard

To estimate the needs and wishes of our users it was crucial to define a target group for TipTop. While sport events cater to a very large group of people from all ages and backgrounds, bets are usually only made between teenagers and older generations. Thus, it was important for us to allow a wide variety of users, from teenagers to elderly people, to efficiently interact and use our application.

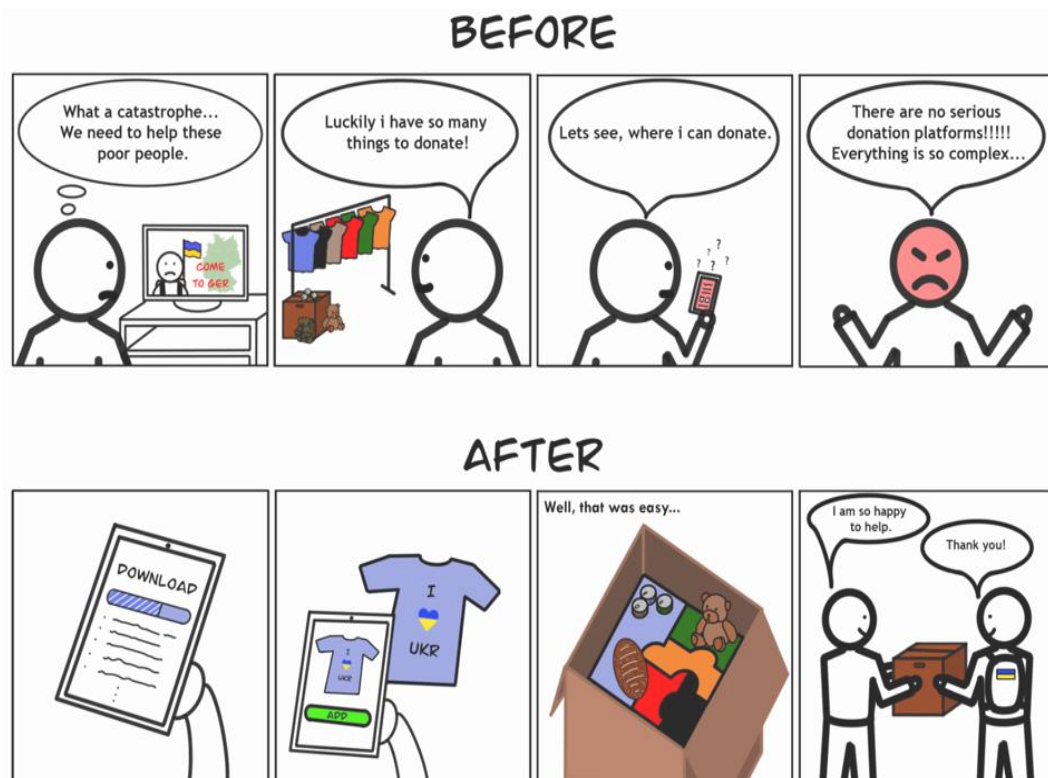


Figure 1 Storyboard

2.2 Personas

Personas allow to gain a better understanding of the needs and restrictions of the user group.

The application has two primary target groups. The first are people from Ukraine who had to leave their country. The second are the residents of Germany who are willing to help as a donor. People with all sorts of different background, age or technical proficiency should be able to use the application. We have created two Personas. Sabine Gutenberg, who represents the target group of donors. Diana Alekseev, who represents the target group of refugees from the Ukraine.

2.2.1 Persona 1

Name: Sabine Gutenberg

Age: 57

Gender: female

Profession: CEO



Figure 2 Persona 1

https://www.t-online.de/gesundheit/krankheiten-symptome/id_81394376/eine-homoneersatztherapie-nach-den-wechseljahren-ist-nicht-ratsam.html

Description: Ms. Gutenberg is the CEO of her own company. She has very little time because of her stressful job. She was shocked when she heard about the situation in Ukraine and the amount of refugees that had to leave their country. Because of that, she decided to help these people with donations. It's very important to her, that the donations will directly reach the people in need. She does not trust organizations because of past incidents thus she would prefer the recipients to personally get the donations from her.

Key characteristics:

- Not much time because of her job
- Open to make new contacts
- Has many items that she is willing to donate
- Wants to help the people who really need it

Goals:

- Wants to be socially engaged
- Wants the donations to reach the recipients

Use case scenario: Miss Gutenberg still has many useful items in her house from her kids that already moved out. For example, clothes, furniture, electronics etc. Most organizations are either too far away, or she distrusts them because they are not reputable enough for her. She would love the recipients of the donations to directly get them from her.

Skills:

Technical skills: 6

Social skills: 10

Language skills: German and English

2.2.2 Persona 2

Name: Diana Alekseev

Age: 39

Gender: female

Profession: Mason



Figure 3 Persona 2

<https://www.unwomen.org/en/news/stories/2020/2/feature-anastasiia-yeva-domani-transgender-woman-fights-for-her-rights>

Description: Miss Alekseev trained to become a mason in Ukraine. Because of the ongoing war, she had to leave her country with her 10-year-old son and her elderly parents. She relies on donations because she has yet to find a job in Germany. She also does not have many possessions because she had to abruptly leave Ukraine. Her main goal is to integrate independently as much as possible in the new country. But this process is hard for her because of communication problems caused by the fact that she can only speak Ukrainian. Before she had to leave, she was a mason in a medium-sized company while living with her husband, kid and her parents in a multigenerational household.

Key characteristics:

- Refugee, has yet to find a job
- Is used to organize anything family related
- Only speaks Ukrainian, thus there is a communication barrier in Germany

Goals:

- Quick integration in Germany
- New contacts
- Needs donations to find a footing in Germany

Use case scenario:

Miss Alekseev left her county because of the sudden war with Russia. She found an apartment to transitionally live there with her family. But because she has no income, many things are missing in their everyday life.

She relies on helpful people from her current surroundings and local organisations. But oftentimes the most needed items are not included in those donations. She would like to search for specific items and is also ready to get them herself while also making new contacts with residents.

Skills:

Technical skills: 5

Social skills: 5

Language skills: Ukrainian

2.3 User Stories

User Stories specify software requirements from the perspective of a user of the application. We defined personae, that represent the typical user role. According to those typical user roles, we have described the requirements each user has. For each of the requirements we have defined, there are acceptance criteria, that have to be met to fulfil the user's needs.

1. "As a refugee, I want to be able to choose the items of the donations by myself because I currently need only certain items."

Acceptance Criteria:

- Refugees can see a list of all offered items.
- The list of offered items is searchable in order to pick the needed item.

2. "As a refugee, I want to be able to make advertisements to tell, for which specific items I'm searching, in order to get potential donors realize what I need. "

Acceptance Criteria:

- Refugees can define items they need and add them to the list of needed items, the donor can see.
- In an advertisement, the refugee can describe in detail, what he needs.

3. "As a refugee, I want the app to support my chosen language because I currently lack any other language skills."

Acceptance Criteria:

- The text of the advertisement can be translated to the language the user has set in the phone settings. If a language is not supported, the default language is English.
- The user interface is available in German, English and Ukrainian. Language is selected according to the phone settings.

4. “As a refugee, I want to be able to see only advertisements I can reach because my transportation capabilities are limited. “

Acceptance Criteria:

- The advertisements shown are only in the same city.
- Definition of a search radius is possible.

5.“As a donor, I want an easy way to offer donations because I can’t invest too much time.”

Acceptance Criteria:

- Donors can define items they offer and add them to the list of offered items, the refugee can see.
- In the advertisement, the donor can describe in detail, what he offers.

5.“As a donor, I want to be able to see what the people actually need for donation, because I want to donate what is really needed.”

Acceptance Criteria:

- Donors can see a list of all needed items
- The list of needed items is searchable to see if items the donor has are actually needed.

6.“As a donor, I want to be able to set whether I want to have the donation be picked up or see where I can drop it so people can pick it up in that location because I don’t have much time. “

Acceptance Criteria:

- In the description of the advertisement, the donor can describe the terms of delivery or pick up.

2.4 Use Cases

Use Cases describes the interaction of a user with the application to reach a specific goal. It is used to document functionalities from the perspective of the user and consists out of a sequence of actions that take place in a prescribed order.

We described our Use Cases with the User Roles and a detailed description.

2.4.1 Search an item

Actors:

- User
- UkrAid app

Use Case Description:

The user selects the home view

The UkrAid app shows all advertisements in the user's location (default only offers)

The UkrAid app provides filter options to select relevant advertisements:

- The User selects the type of the advertisement (offer or search)

The UkrAid app shows only advertisements that have the selected type

- The User types a keyword in the search field.

The UkrAid App shows only those advertisements that match the keyword.

- The User selects a category in the category scroll bar.

The UkrAid App shows only those advertisements that have the selected category

The user can select an advertisement

The UkrAid App displays the detail view page

2.4.2 Create an advertisement

Actors:

- User
- UkrAid app
- Gallery app (Android standard gallery app)

Use Case Description:

The user selects “+” icon to create an advertisement

The UkrAid app shows Create Advertisement form

The user has to choose whether it is an offer or a search.

- If the user has selected “I search”, the UkrAid app hides the “add photo” button because searches do not contain photos.
- If the user has selected “I offer”, he can use the add photo button to add a photo to the advertisement. In this case:

The UkrAid app opens the phone’s gallery app.

The user selects one or multiple photos in the gallery app and confirm the choice.

The UkrAid App shows a scrollable preview of the selected photos.

- If the user wants to remove a picture, he can click on the “x” icon
- In this case the UkrAid app removes the photo from the scrollable preview

The user has to type a title in the title field. (mandatory)

The user has to select a category out of a list of predefined categories (mandatory)

The user has to type a location in the location field (mandatory)

The user has to type a text in the description field (mandatory)

The user clicks on “advertise” to make the advertisement.

The UkrAid app checks whether all of the mandatory fields contain correct data. If not, an error message is displayed. The user can then complete/correct his input. If everything is correct, the UkrAid App stores the advertisement in the database.

The UkrAid App displays the main screen.

2.4.3 Contact advertiser

Actors:

- User
- UkrAid-app
- E-Mail app (Android standard E-Mail app)

Use Case Description:

The user selects an advertisement.

The UkrAid app shows the detail view of the advertisement.

In this view UkrAid app provides a button to contact the advertiser.

The user clicks on the contact button.

The UkrAid app opens the send form in the e-mail app.

The UkrAid app fills in the following fields:

“To” field: E-mail address of the advertiser

“Subject” field: Title of the advertisement

“Message” field: Pre-defined sample text

The user can modify sample text in the e-mail app.

The user selects “send” in the e-mail app to contact advertiser

2.4.4 Manage advertisements

Actors:

- User
- UkrAid app

Use Case Description:

The user selects Profile View

The UkrAid app shows all active advertisements of the user

The user can click on the action button.

To manage advertisements

The UkrAid app displays a popup with options to choose:

- The user can select “cancel”

The UkrAid app closes the popup

- The user can select “edit”

The UkrAid app shows an Edit Advertisement screen

The user can change the contents of the advertisement

The user clicks on “update” to update the advertisement.

The UkrAid App checks whether all of the mandatory fields contain correct data.

If not, an error message is displayed. The user can then complete/correct his input.

If everything is correct, the UkrAid app stores the modified advertisement in the database.

- The user can select “delete”

The UkrAid app displays a confirm dialogue If the user confirms the deletion of the advertisement

The UkrAid app deletes the advertisement in the database If the user selects “cancel”

The UkrAid app closes the popup, the advertisement is not deleted from the database

2.5 Requirement Specification

From the definitions of the Personae, User Stories and Use Cases, we defined functional and non-functional requirements. Every requirement consists of a name, ID, description, success criteria and the priority. We grouped them into “Must have requirement” and “May have requirement” to priorities the requirements for the planning and development process.

2.5.1 Functional Requirements

Table 1 Functional requirement – must have – Registration user

Requirement	Registration of new User
ID	F1
Description	An unregistered User should be able to create an account and register for using the app. The new user should type in his real name, his e-mail-address, and a password.
Success criteria	When the user has successfully created an account, he has access to the functions of the app. The user data is stored in the DB.
Priority	Must have requirement

Table 2 Functional requirement – must have – Login

Requirement	Login of existing User
ID	F2
Description	A registered user should be able to get access to the app after typing in his e-mail address and password
Success criteria	When the user has typed in the correct authentication data, he has access to the functions of the app. If authentication fails, an error message appears.
Priority	Must have requirement

Table 3 Functional requirement – must have – Editing

Requirement	User Profile Editing
ID	F3
Description	An existing user should be able to change his name, his location and add a profile picture.
Success criteria	When the user changes his data, the new values are stored in the db.
Priority	Must have requirement

Table 4 Functional requirement – must have – Creation new Ad

Requirement	Creation of new Advertisement
ID	F4
Description	The User can create an Advertisement. It should be possible to define whether the advertisement is an offer or a search. Offers and searches contain a title, a category, a location, and a description field where details can be described. In Offers Pictures are also contained.
Success criteria	When the User opens the Create Advertisement screen, he has to choose whether the new advertisement is a search or an offer. He can type the title, choose a category and the location. He can also make a detailed description. When submitted, the data is stored in the database. The user can see the advertisement in the list of advertisements in his user profile.
Priority	Must have requirement

Table 5 Functional requirement – must have – Read ad

Requirement	Read advertisement
ID	F5
Description	The User can select an advertisement and show the content.
Success criteria	When a user clicks on an advertisement, a screen appears that shows title, category, details and the name of the advertiser.
Priority	Must have requirement

Table 6 Functional requirement – must have – Edit ad

Requirement	Editing Advertisements
ID	F6
Description	The User can edit his advertisement and change the title, category, description
Success criteria	When the user selects Edit Advertisement, a screen appears, where he can change title, category and description. The new values are stored in the database.
Priority	Must have requirement

Table 7 Functional requirement – must have – Delete ad

Requirement	Delete Advertisements
ID	F7
Description	The user can delete his advertisement
Success criteria	When the user selects Delete Advertiser, a Popup appears, where the user can confirm the deletion. The advertisement is then deleted from the database and is no longer shown in search lists and in the user's profile.
Priority	Must have requirement

Table 8 Functional requirement – must have – Search ad

Requirement	Search for Advertisements
ID	F8
Description	The User can select advertisements of interest by searching in the titles of the advertisements
Success criteria	When the user types in letters in the search textfield, the advertisements that contain the typed in letters in the title are shown. If there is no match, the display shows "no items found"
Priority	Must have requirement

Table 9 Functional requirement – must have – Filter ads by category

Requirement	Choose category of interest to filter results
ID	F9
Description	The User can select a category of interest from a list of predefined categories to select advertisements of interest
Success criteria	A user can select predefined categories from a scroll bar to show only the advertisements of the selected category. If no advertisements match the selected category, the display shows "no items found"
Priority	Must have requirement

Table 10 Functional requirement – must have – Pick gallery images

Requirement	Picking Photos from Phone Gallery
ID	F10
Description	The user should be able to select pictures that are contained in the phone's gallery and enter them as pictures in the user profile and for the advertisements (only offers).
Success criteria	When a user clicks the "add photo(s)" symbol, the phone gallery is shown, where he can select one or multiple photos for addition in either the user profile or the advertisement. If confirmed, the photos are shown in the dialog.
Priority	Must have requirement

Table 11 Functional requirement – must have – Categorize items

Requirement	Categorizing of Items in Advertisements
ID	F11
Description	The User has to choose one of the predefined Categories when creating an advertisement.
Success criteria	When the user creates an advertisement, the selection of a category is mandatory, because every advertisement should be searchable by category.
Priority	Must have requirement

Table 12 Functional requirement – must have – Filter ads by location

Requirement	Advertisements that are not in the User's area are hidden (Advertisements are only locally searchable)
ID	F12
Description	Advertisements do only match if the donor and the receiver can meet (in the same area)
Rationale	
Success criteria	When a user searches through advertisements, only those advertisements are shown where the location of the user and the location of the advertisements are the same. This prevents the user of seeing advertisements that are not relevant because they are too far away.
Priority	Must have requirement

Table 13 Functional requirement – must have – Filter ads by type

Requirement	Choose type of advertisement to filter results
ID	F13
Description	The User can select if he wants to see offers or searches
Success criteria	A donor may only see which items are searched, a refugee may only be interested to see which items are offered. Therefore, the user can choose the type of advertisement he wants to see by a dropdown menu.
Priority	Must have requirement

Table 14 Functional requirement – must have – Show own ads

Requirement	Show overview list of own advertisements
ID	F14
Description	In his user profiler, the user is shown a list of his own advertisements to have an overview and to choose of for editing and deletion.
Success criteria	When the User selects his user profile, a screen appears where a list of all his active advertisements are shown.
Priority	Must have requirement

Table 15 Functional requirement – must have – Contact advertiser

Requirement	Contact advertiser
ID	F15
Description	In the detailed view of an advertisement, a user can contact the advertiser (open e-mail app)
Success criteria	When a user opens the de detailed view of an advertisement a contact button is shown. When clicked the user is redirected to the e-mail-program, where a new message is created, the “to” field is filled with the e-mail address of the advertiser, the subject field is filled with the title of the advertisement.
Priority	Must have requirement

Table 16 Functional requirement – may have – Translation of ads

Requirement	Translation of Advertisements
ID	F16
Description	The User can translate the title and the description of an advertisement to the language selected on the smartphone
Success criteria	When the user has opened the detailed view of the advertisement, he can select the original language of the advertisement or the translation (to the language the app is set to)
Priority	may have requirement

Table 17 Functional requirement – may have – Camera function in app

Requirement	Pictures can be taken in the app
ID	F17
Description	The user should be able to take pictures with the phone camera and enter them as pictures in the user profile and for the advertisements (only offers).
Success criteria	When a user clicks the “camera” symbol, the phone camera, where he take a photo for addition in either the user profile or the advertisement. If confirmed, the photos are shown in the dialog.
Priority	may have requirement

Table 18 Functional requirement – may have – Geofencing

Requirement	Geofencing to calculate radius search
ID	F18
Description	The User can define a radius to define the area of search
Success criteria	When a user searches through advertisements, only those advertisements are shown where distance between the location of the user and the location of the advertisements within a defined radius. This prevents the user of seeing advertisements that are not relevant because they are too far away
Priority	may have requirement

Table 19 Functional requirement – may have – Roles for users

Requirement	Specific roles (drivers, organizations) for certain users
ID	F19
Description	In order to implement some advanced features (e.g. logistic functions), a user should be able to select specific user roles (e.g. drivers to pick up goods from donators to bring them to refugees)
Success criteria	When a user is created, a specific user role can be chosen. When confirmed, the role is stored in the database. Certain advanced features may be available for those users.
Priority	may have requirement

Table 20 Functional requirement – may have – Direct message

Requirement	Direct message functionality
ID	F20
Description	The user can communicate directly with the advertiser in the app by a chat function. Offers more comfort.
Success criteria	When a user opens the de detailed view of an advertisement a “chat” button is shown. When clicked, a chat screen opens, where the user can communicate with the advertiser by a chat function within the app. The chat screen can be selected at any time in the app.
Priority	may have requirement

Table 21 Functional requirement – may have – Real time translation

Requirement	Real time translation of messages
ID	F21
Description	Within the Chat function of the app, the user can type and see messages in his own language. An automatic translation enables the other chat partner to communicate in another language.
Success criteria	When a user sends a message in the chat screen, the message is automatically translated to the language of the receiver. The receiver also gets the original message in brackets.
Priority	may have requirement

Table 22 Functional requirement – may have – User notification

Requirement	Notification of users when an event happens (offers matching, chat message received)
ID	F22
Description	Users should be able to get notified, when an action of interest in the app happens.
Success criteria	The User can define searches / categories and is automatically notified when a new advertisement is placed that matches the filter definition. When a Chat message is received, the user also gets a notification.
Priority	may have requirement

2.5.2 Non-Functional Requirements

Table 23 Non-functional requirement – must have – OS

Requirement	Primary for Android
ID	NF1
Description	The App should be able to run on Android OS
Success criteria	The app should run on Android OS without any trouble
Priority	must have requirement

Table 24 Non-functional requirement – must have – Localization

Requirement	App supports localization
ID	NF2
Description	The app should be able to adapt to the user's language settings.
Success criteria	The app uses the predefined language of the smartphone. If not supported, the default language is English.
Priority	must have requirement

Table 25 Non-functional requirement – must have – Response time

Requirement	Low response time (<1s) while using the app
ID	NF3
Description	Whenever the user clicks within the app, the user will be redirected to the next screen within one second.
Success criteria	By checking each screen, a response time of less than a second is achieved.
Priority	must have requirement

Table 26 Non-functional requirement – may have – Update cache

Requirement	Update cache in an interval of 5 minutes
ID	NF4
Description	The app should be able to store data in the cache in order to have access to a local database if the quality of the mobile internet is poor.
Success criteria	If the user has instable internet connection, it should be possible to store data to a cache. It should be updated at least every 5 minutes.
Priority	may have requirement

Table 27 Non-functional requirement – may have – Availability

Requirement	Availability of the app needs to be secured
ID	NF5
Description	The App should be available most time.
Success criteria	Our App use Firebase which has an uptime of 99.95% of the time. Means the App should also has at least the same availability.
Priority	may have requirement

Table 28 Non-functional requirement – may have – Security of data

Requirement	Security of data
ID	NF6
Description	To keep the data secured, a secure database system has to be used.
Success criteria	The app uses firebase, in order to get a secure data storage.
Priority	may have requirement

3 Conceptual model

The basis of our development is the conceptual model. It was created to describe and visualize the application and its functions on paper. Different diagrams and especially Mock-up's were used for this step.

3.1 Component diagram

A Component diagram shows the components, interfaces, ports and relationships between the elements. UkrAid is a mobile application, meaning it runs on smartphones like Android or iOS devices. These devices are basically the clients that make calls to Firestore which is our backend.

We use three different functionalities of Firestore in Firebase Authentication, Firebase Cloud Storage and the Cloud Firestore.

UkrAid is a multiuser application, thus we need to handle the different user profiles. Firestore makes this relatively easy by use of the Firebase Authentication. With this we can handle all our profiles.

Advertisements obviously need to be persistently saved, meaning that a database is required. We could have used for example a MySQL database, but Firestore already offers the Cloud Firestore and Cloud Storage. This means we had no need for an additional backend that connects to Firebase just for the authentication but could also use the Firestore database which in turn makes it our complete backend. Cloud Firestore is a complete NoSQL database used to save our user specific profiles and the individual advertisements. Cloud Storage is used to save specific files. In our Use-Case we save the pictures of the advertisements there to later have a reference in our corresponding ad.

Additionally, we have implemented a translate function which makes an API call to LibreTranslate. This is an open-source Machine Translation API which helps us to identify and translate certain texts.

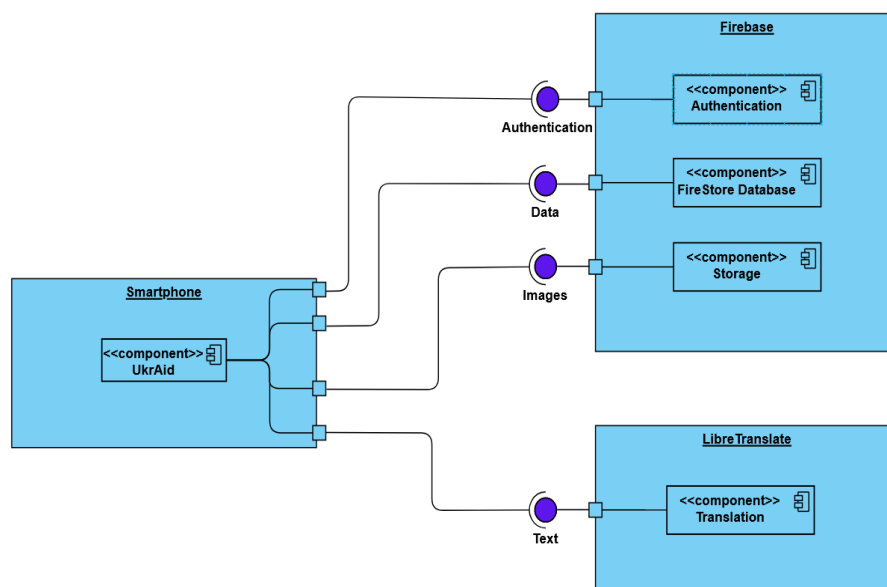


Figure 4 Component diagram UkrAid

3.2 Sequence diagram

Sequence diagrams are interaction diagrams that show how operations are carried out. These diagrams are time focused and show the order of the interactions visually.

The UkrAid application has many different interactions. The following are two examples of the sequence of creating a user and creating an ad.

3.2.1 Create User

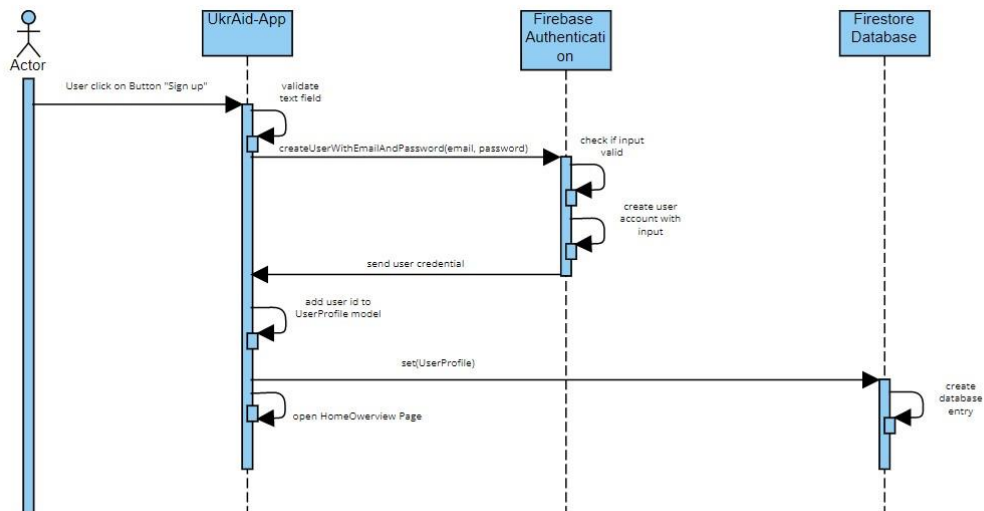


Figure 5 Sequence diagram Create User

After the user has filled out the signup form and clicking on the “Sign up” button, the application will create a user with the Firebase Authentication based on email and password that has been given by the user. After having successfully created the user, the application will make an entry in the database based on the UserProfile model that was given by the app and the userID (field in UserProfile class) that was registered in the Firebase Authentication.

After the successful creation of the user in the database, the user will be redirected on the “HomeOverview” page.

3.2.2 Create Ad

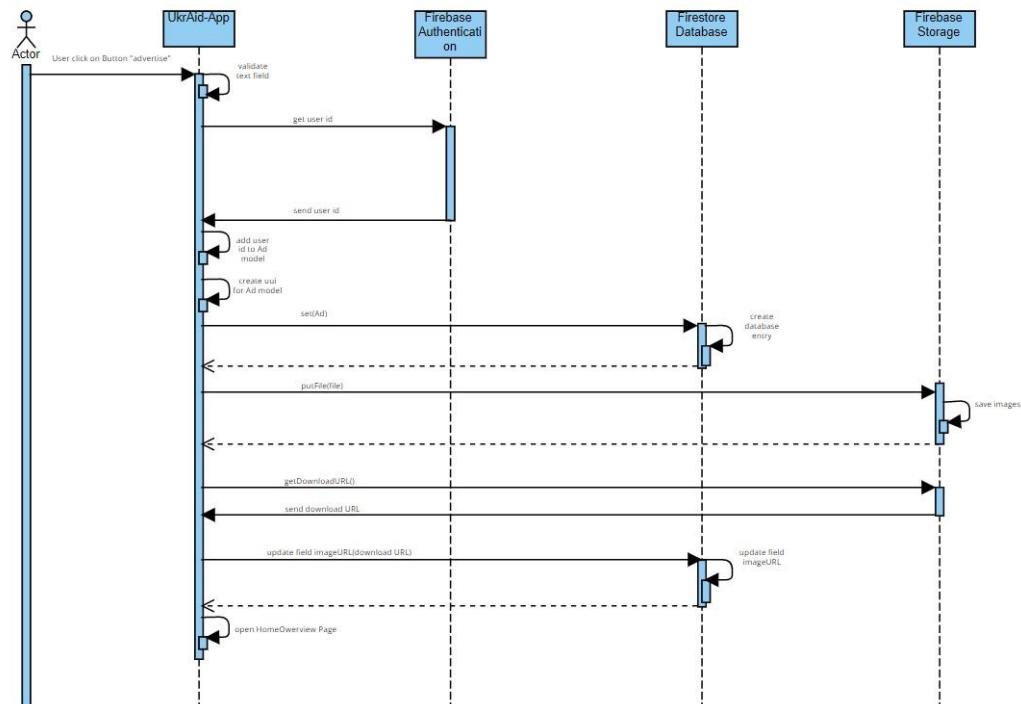


Figure 6 Sequence diagram Create Ad

Creating an Ad is similar to creating a user at least in some aspects. After the user has filled out the necessary fields to create an ad, the ad model will be filled with the data and userID just like the UserProfile. But additionally, the ads need some kind of primary key to get distinguished. Thus, they also get a UUID to achieve this individuality.

The created ad will then be stored in the Firestore Database. The major difference is in the case the ad also includes images. These images need to be stored in the Storage of Firebase and then referenced in the specific ad entry in the database. This can include more than one image. After having created the ad, the user will be redirected to the "HomeOverview" page again.

3.3 Mock-Ups

Mock-Ups are used to visualize the application before actual implementation.

This helps to achieve a unified design across the entire application as well as discuss the user experience for each screen.

We designed the following Screens with the Adobe XD Tool and decided to create High-fidelity Mock-Ups to exactly know how the implementation will have to look like.

3.3.1 Login Screen

The Login Screen is the first screen of the application that is shown, when a user starts the application. When user already has an account, he can complete his user credentials or if the user doesn't have an account, he can enter the Sign-up Screen by using the Sign-up button.

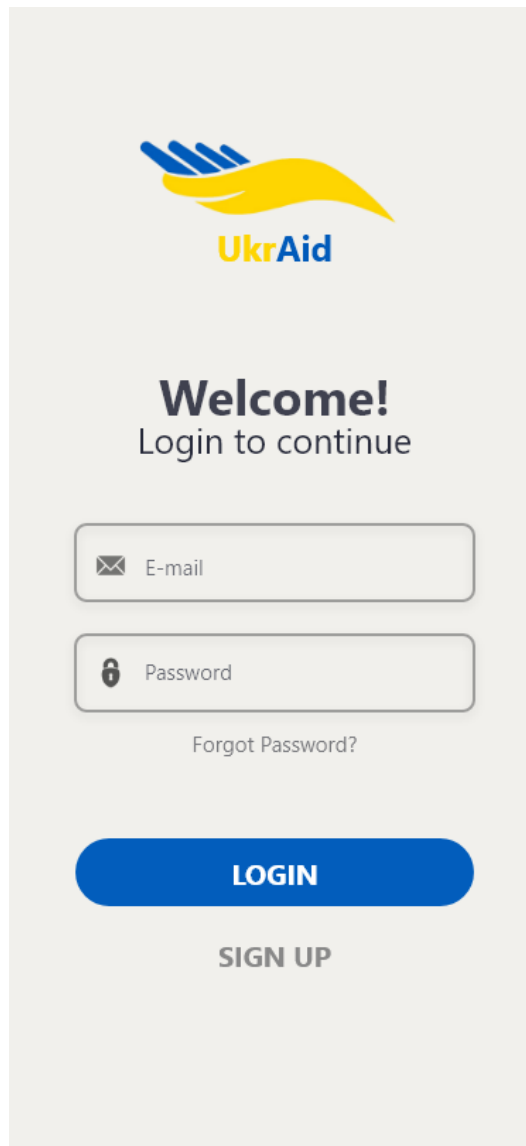


Figure 7 Login Screen

3.3.2 Sign up Screen

This Mockup Screen shows the Sign Up-Screen. If the user doesn't have an account, he can create an account by filling in the text fields. Here we have the text fields "Name", "E-Mail", "Password" and "Confirm Password". The user has to fill in all text fields to create an account.

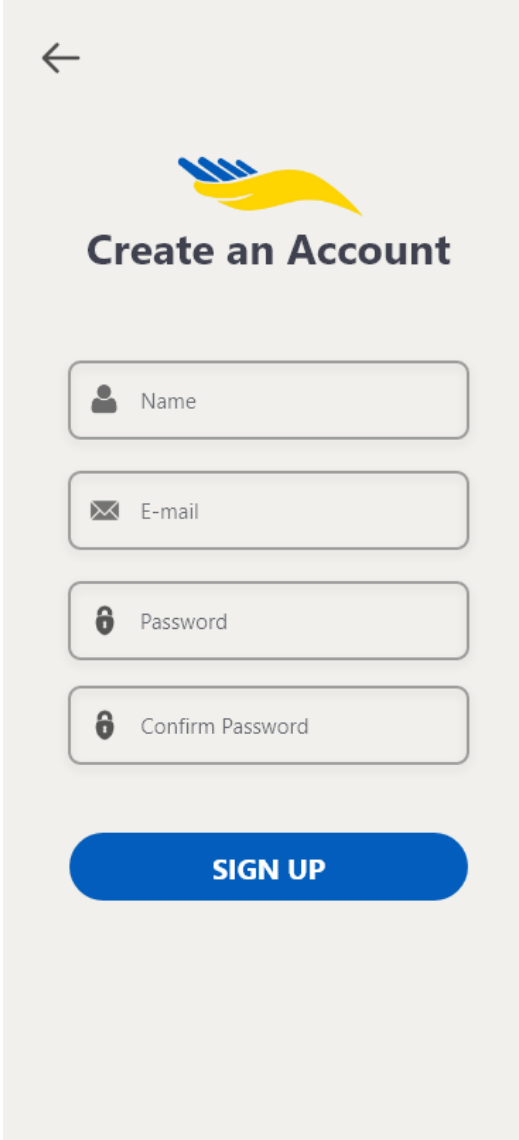
The image shows a mobile app mockup for a sign-up screen. At the top left is a back arrow. Below it is a logo consisting of a blue hand holding a yellow flame. The title "Create an Account" is centered below the logo. There are four text input fields stacked vertically: "Name" with a person icon, "E-mail" with an envelope icon, "Password" with a lock icon, and "Confirm Password" with a lock icon. At the bottom is a blue rounded button with the text "SIGN UP" in white capital letters.

Figure 8 Sign-Up Screen

3.3.3 Home Screen

This is the Home Screen of the application. This Screen is shown to the user after having successfully logged in or signed up. When the user is in the app, he can get back to this screen by clicking the “Home” icon. The Home Screen shows the advertisements that are available in the user’s location. By default, the “Offer” advertisement type is selected. The user has several filter options to find the right advertisement. First, he can select the advertisement type (“Offer” and “Searches”) in a dropdown. By selecting from a scrollbar of categories, he can pick a specific category. In a search field, the user can search for keywords in the advertisement titles. Each advertisement that is listed, shows a picture if a picture is part of the ad, the title and the location of the advertisement. By clicking on an advertisement, the user can see the selected advertisement in detail.

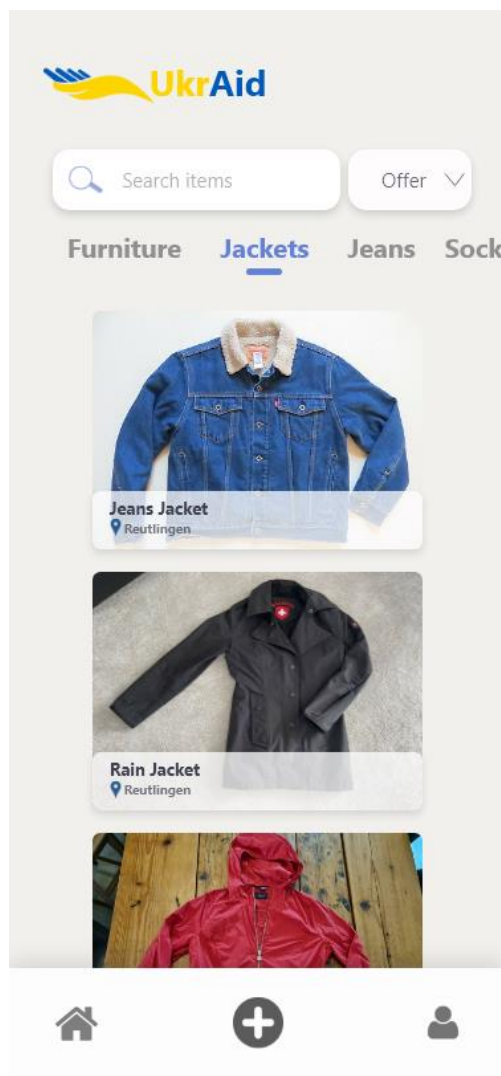


Figure 9 Home Screen

3.3.4 Detail Advertisement Screen

This Mock-up Screen shows the Detail Screen of an advertisement. After selecting an Advertisement in the Home Screen, the user can see the details of the advertisement. Here all the pictures of the ad are displayed. By scrolling to the right, the user can see the other ones. Here, the title, the location, the advertiser's name and the description of the ad are shown, too.

When the user is interested in the ad, he can write a message into the text field and can contact the advertiser.



Figure 10 Ad Detail Screen

3.3.5 Create Advertisement Type Offer Screen

This Mock-Up screen is the Create Advertisement screen for the advertisement type “Offer”. When the user clicks on the “+” Icon in the Navbar, this screen appears. Here the user can create an ad with the ad type “Offer”. By clicking on the Photo Icon, the user can add pictures to the advertisement. The selected ones are shown on the screen. The user can type in the title, location and the description of the ad and select a category from a dropdown. All text fields are mandatory except the pictures.

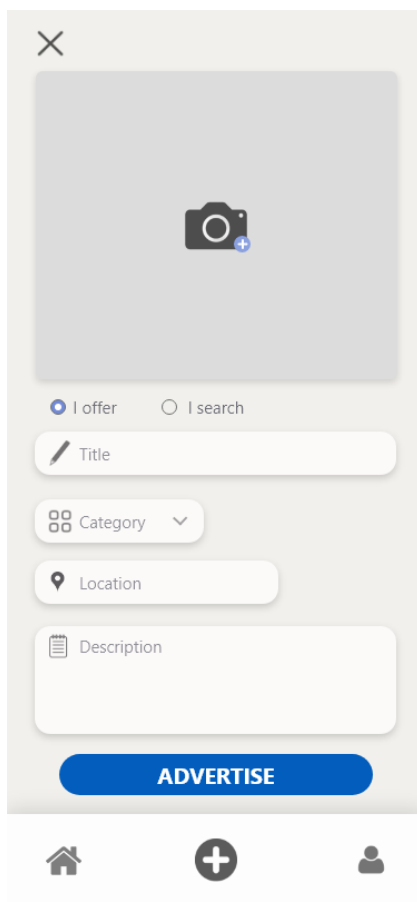


Figure 12 Ad Create Screen

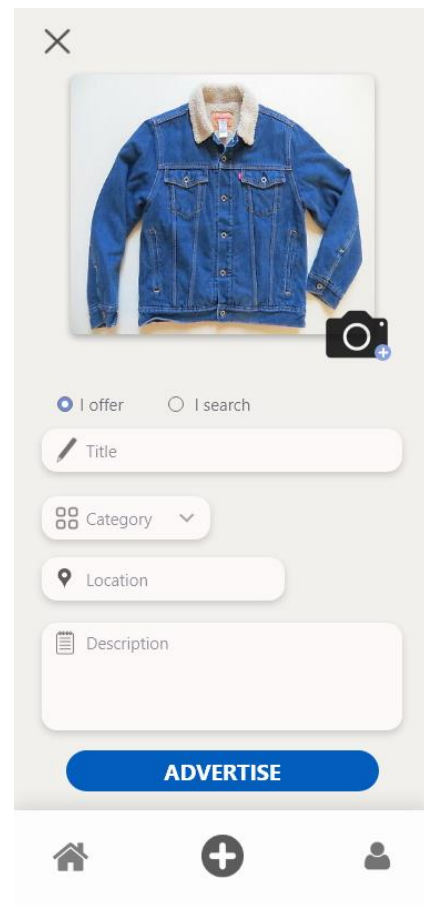


Figure 11 Ad Create Screen
with Picture

3.3.6 Create Advertisement Type Searches Screen

This Mock-up screen is the Create Advertisement screen for the advertisement type “Searches”. This Screen appears when the user selects “I search” in the Create Advertisement Screen. The user can create an advertisement with the type “Search”. In this case the user can’t add pictures. The user can type in the title, location and the description of the ad and select a category out of a dropdown. All text fields are mandatory.

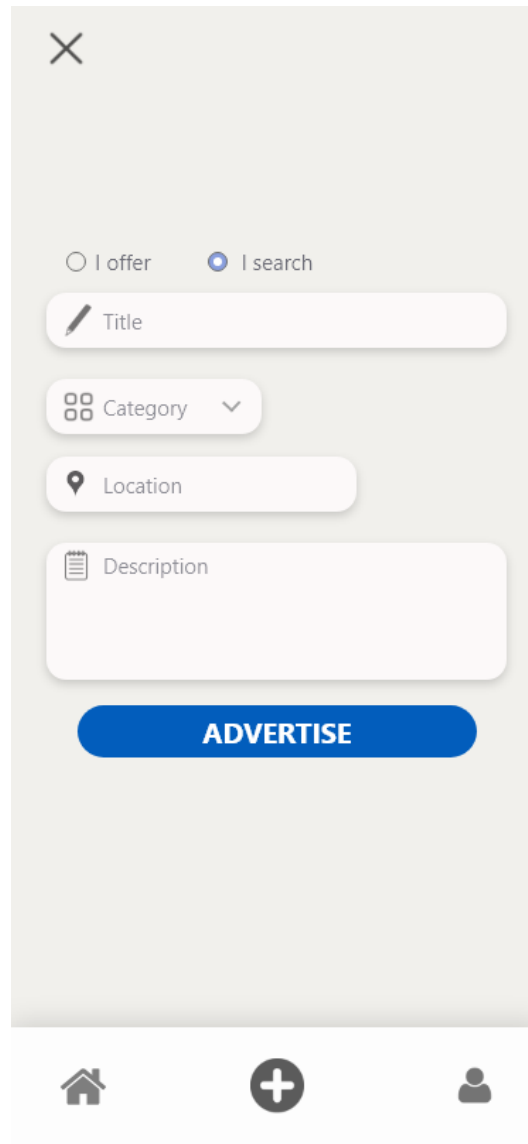


Figure 13 Ad Create Screen search

3.3.7 Profile Overview Screen

This Mock-up screen is the user Overview Screen. The screen appears when the user clicks on the “Profile” Icon in the NavBar. In this Screen the user can see all active advertisements he has created and his own user information. By clicking on the “...” Icon within an ad, the user can edit or delete the selected advertisement. The “Given Away” Button can be used to deactivate the advertisement to reserve the ad. By clicking the Settings Icon, the user can edit his user information.

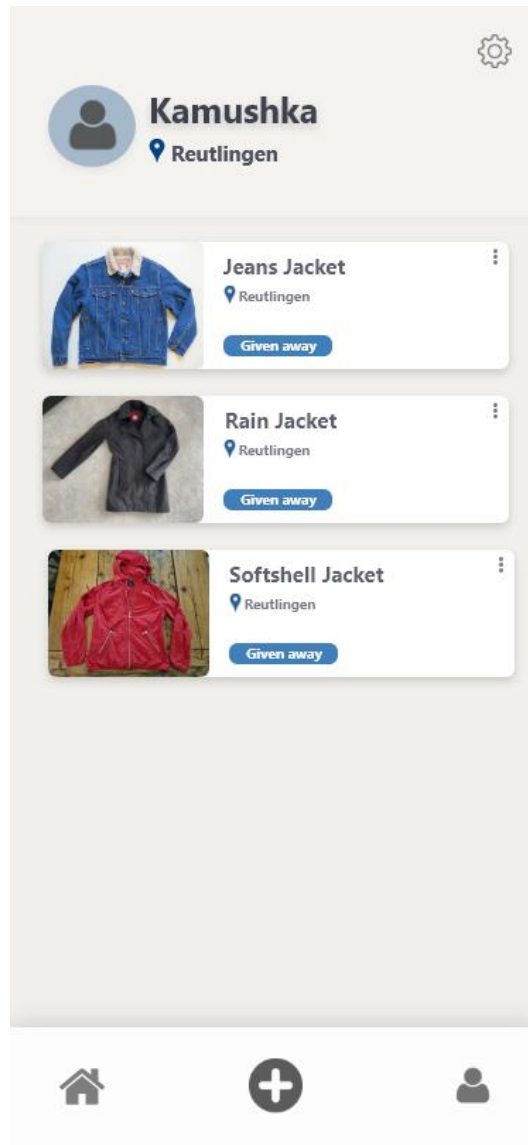


Figure 14 Profile Screen

4 Software Architecture

UkrAid is implemented with the Flutter framework and the language Dart. To make the code maintainable and easily understandable for others, these patterns also make the software development process quicker and more efficient thus resulting in a higher quality of the application.

4.1 Layered Pattern

The layered Pattern, or n-tier architecture, is one of the most basic patterns there is. The Idea is that the code is separated into different layers. This includes for example a Presentation Layer, Business Layer, Persistence Layer and Database Layer. A Request must basically go through every single one of these layers. The Advantage is that this pattern is simple and easy to understand while also reducing dependencies because the function of each layer is separated from the others. A disadvantage of this pattern architecture is the difficulty of changing the architecture later.

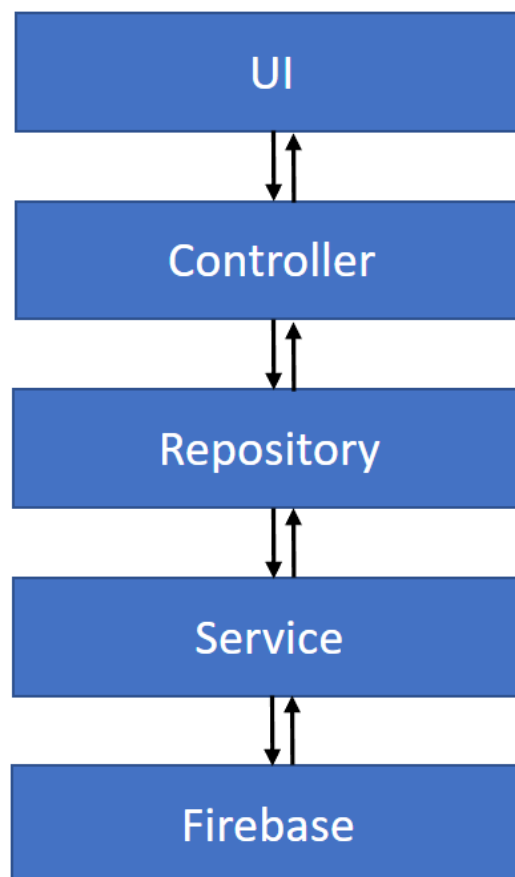


Figure 15 Layerd pattern UkrAid

The architecture in UkrAid follows the architecture depicted in Figure 15. The UI layer is for the user input and separated into screens. Each screen also has a designated controller so the business logic can be implemented there instead of complicating the UI logic classes.

The controller then has access to the repositories. This is where domain objects, for example “ad”, get converted into data transfer objects (DTOs) and back. The DTO then gets passed to the individual service which in turn can use these objects to make the call to the Firestore database. The result then needs to be passed back to every layer again back to the UI layer.

4.2 MVC Pattern

The MVC pattern is one of the most popular patterns in frontend designs. The three classes separate the three areas of responsibility. The controller has the entire control of the business logic, while the view is responsible for the GUI the user can see. The model is the structure that contains the data and gets presented in the view. Changes in the data will oftentimes be presented with the observer pattern implemented with the view. Sometimes the model also contains some business logic.

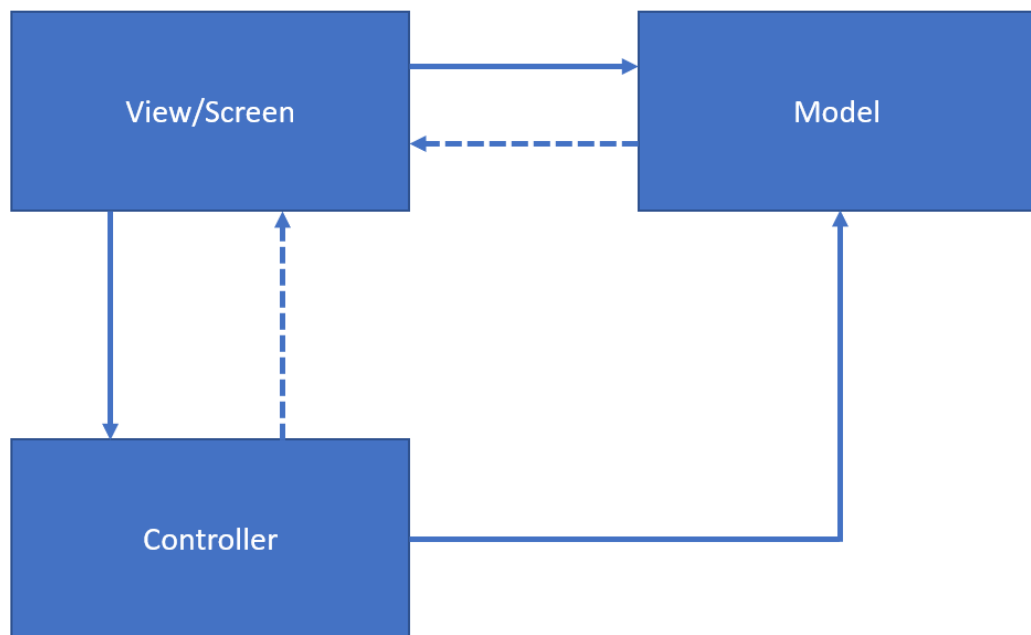


Figure 16 MVC pattern

The MVC pattern in the UkrAid application is not implemented like in literature. For example, the models do not have implemented any business logic, but are just normal objects. The view presents the GUI to the user, while using the model to map the input. This can then be given to the controller that handles the business logic that mostly includes giving the data to the repository layer in this case.

We had to use the dependency “mvc_pattern” version 8.10.2 to implement the MVC pattern into the project. This dependency handles the connection between each abstraction. The package was made by andrioussolutions.

Every single screen has a corresponding controller that can be used by the screen itself. This separation helps the UI portion of the code to be more clean and easily understandable.

4.3 Repository Pattern

The Repository pattern is basically an extra abstraction in the database layer. This pattern prevents business logic to contain any sort of direct access to the database. Having the business layer make the API/database calls directly would basically mean the same code repeating itself throughout the application. With the Repository pattern we get a central access Point for the requests.

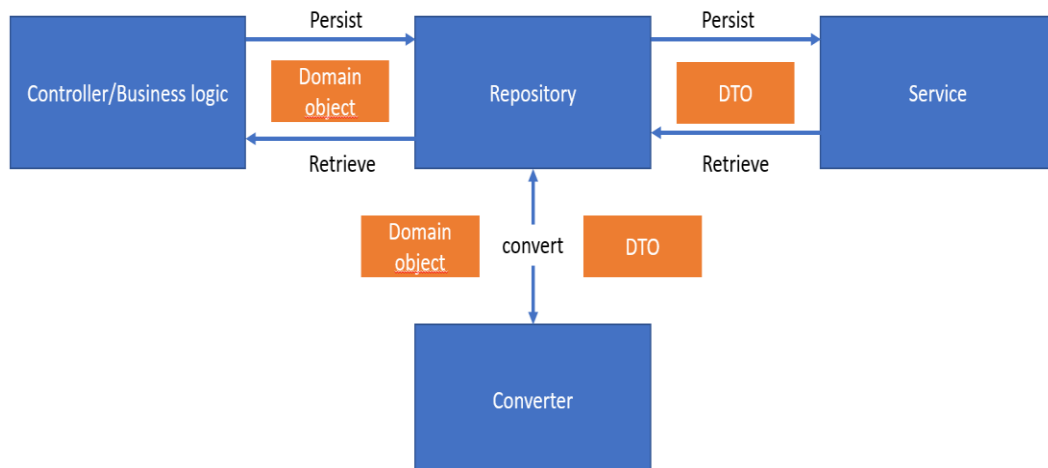


Figure 17 Repository pattern UkrAid

As seen in the Figure 17, the controller can make a call to persist a certain domain object, passing it to the responsible repository. The Repository then takes this object, and uses a converter, which is basically a mapper, to get the needed DTO. This DTO can then be given to the service, which in turn makes the call to the database/API.

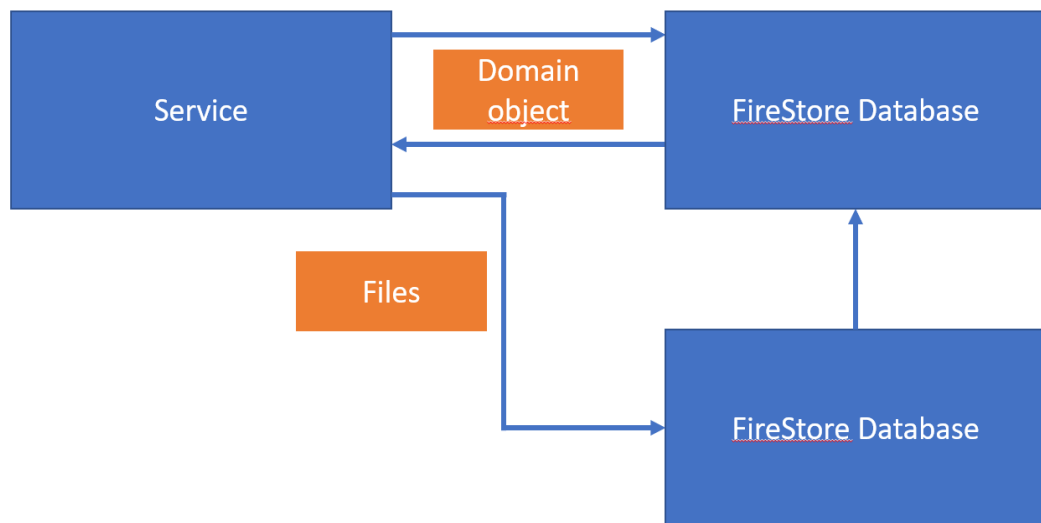


Figure 18 Repository pattern services UkrAid

These DTOs are necessary because they guarantee the contract between our client and Firebase backend. The domain objects are now independent from the DTO objects, meaning the domain objects can be changed, if necessary, without accidentally hurting the data structure in the database.

4.4 Dependency Injection

Dependency injection is basically just injecting the needed dependency into objects instead of having it construct themselves. The three most common injection methods are constructor injection, property injection and method injection. Dependency injection is useful for loose coupling, central initialising and provides huge benefits in testing.

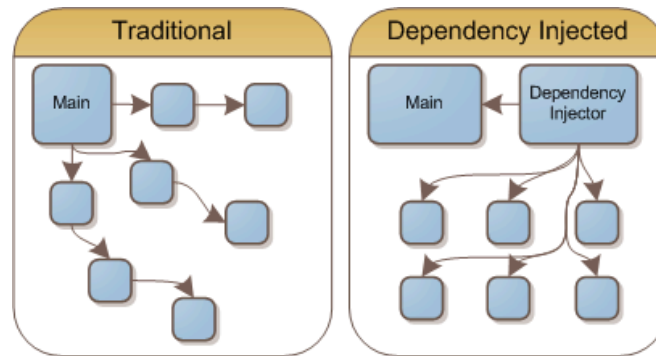


Figure 19 Dependency Injection in general

<https://devopedia.org/dependency-injection>

Flutter does have an out of the box solution of using dependency injection with “inherited widgets” or “provider”, but we decided to use the package “get_it 7.2.0”. The advantage of this package is that it is extremely fast and very intuitive.

The “get_it” packages enable us to initialize our services in the main class, meaning that every service will be ready to use after the app start.

```
final locator = GetIt.instance;

void setupGetIt() {
  ///Services
  locator.registerLazySingleton<AdService>(() => AdService());
  locator.registerLazySingleton<UserService>(() => UserService());
  locator.registerLazySingleton<AuthService>(() => AuthService());
  locator.registerLazySingleton<TranslatorService>(() => TranslatorService());

  ///Repositories
  locator.registerLazySingleton<UserRepository>(() => UserRepository());
  locator.registerLazySingleton<AdRepository>(() => AdRepository());
}
```

Figure 20 Dependency Injection setUp in UkrAid

Every new service/repository can be easily initialized in the function “setupGetIt()” and be accessible throughout the entire application if needed.

4.5 Dependencies

Table 29 Used dependencies

Package Name	Version	Description
flutter_localizations	NA	Internationalization of the app. Needed to have the app display different languages.
intl	0.17.0	
mvc_pattern	8.9.0	Model View Controller pattern implementation.
multi_select_flutter	4.1.2	Creation of multi-select widgets.
smooth_page_indicator	1.0.0+2	Customizable animated page indicator.
image_picker	0.8.5+3	Enables picking images from the device's library.
http	0.13.4	Library for making and consuming HTTP requests.
cupertino_icons	1.0.2	Asset repository containing a default set of icons.
firebase_core	1.15.0	Flutter plugin to use the Firebase Core API, enabling connections to Firebase apps.
firebase_auth	3.3.16	Flutter plugin to use the Firebase Authentication API.
get_it	7.2.0	Simple locator for Dart/Flutter projects enabling a simple way of dependency injection.
cloud_firestore:	3.1.14	Flutter plugin to use the Cloud Firestore API.
firebase_storage	10.2.15	Flutter plugin to use the Cloud Firestore API.
file_picker	4.5.1	Package that allows the usage of the native file explorer to pick a single or multiple files.

uuid:	3.0.6	Simple and fast generation of RFC4122 UUIDs.
url_launcher	6.1.2	Flutter plugin for launching a URL.

4.6 Database

The Firestore database is a NoSQL database. This means there are no tables, but a folder structure. This means the data modelling is pretty flexible and basically schema-less. Firestore runs on a collection -> document structure. A collection is basically a domain that groups certain sets of documents.

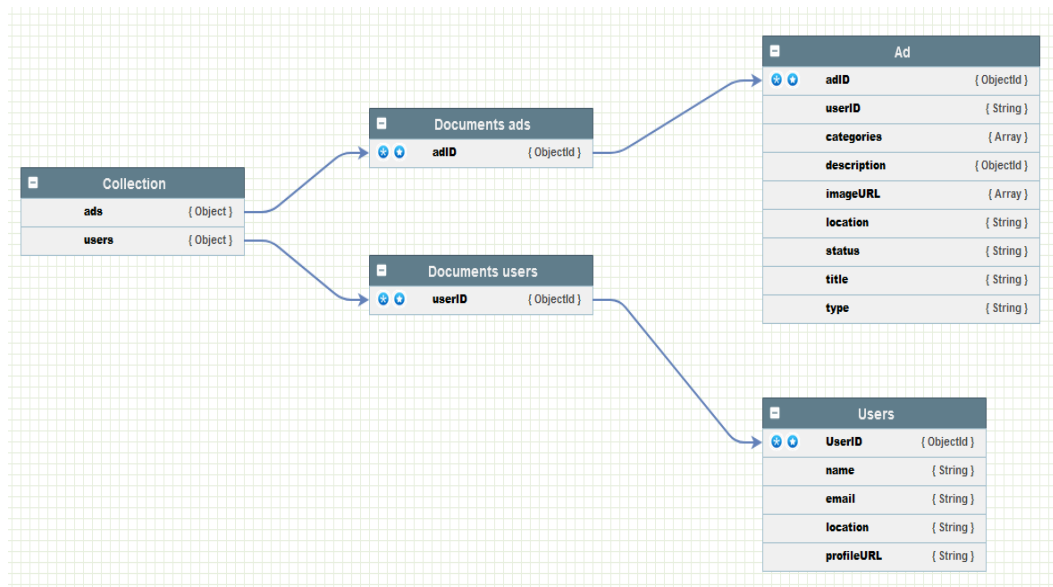


Figure 21 Database model UkrAid

The data model of the UkrAid app has two basic domains. The users who are registered in the Authentication and the ads they create. Each user and every ad have an individual id that can be seen like a primary key. These keys help us to identify the ad or user.

Additionally, the ad also saves the userID of the user who created them. Thus, we can establish a one-to-many relationship between a user and his created ads. Normally this duplication of data is not recommended, but because NoSQL databases are way more flexible than SQL databases, the problem with duplication is rather minor.

To save images, Firebase offers the storage service. This storage also has a folder structure but is only for saving files. To save our images, the application first needs to create an entry in the storage, and then reference the download URL in the specific ad that is supposed to have these images.

The requests to the database need multiple calls to make this happen, because of this the operations need to also be defined in an atomic transaction to secure that the entire process of first storing the files in the storage and then reference it in the database doesn't fail at one point. An undetected failure at any point could lead to data inconsistency.

4.7 LibreTranslate

For the use of the App, we wanted to have a translation option for the advertisements to make it easier for the user to understand what is provided. To avoid expenses, we looked for a free translation API, most of the found APIs was limited in use or had the problem that we had to enter our billing information.

We found the free and Open Source Translation API "LibreTranslate" (<https://github.com/LibreTranslate/LibreTranslate>), that is licenced under the GNU Affero General Public License v3.0. The API can be self-hosted on sever but because we didn't have one, there are some LibreTranslate instances that could be used. For our current use we decided to use the "libretranslate.de" instance for translation.

The request we send to the API is a HTTP-POST Message with an JSON Body. In the body you can specify the language of the text or you can set if the language should be auto recognized and in which language it should be translated.

We used the option of auto translate because we didn't save in which language the advertisement has been created, the language in which the text is translated is the language the app has detected and uses.

5 Result

5.1 Features

5.1.1 Signup and Login Process

The User can login with his user credentials (email and password). For the authorization we used firebase authentication, that checks if the user is already registered.

The user can, as described in the mock-ups and requirements, create an account in the app. In difference to the mock-up the user will also have to set his location in the signup process. This is needed because we have a location search for the place where the user is located. All text fields get checked by typing in, all text fields have to be filled in. The password must have more than 6 characters and the confirmation password has to be equal to the password. The email address should be a valid email address. After the user has signed up, he comes to the home view of the App.

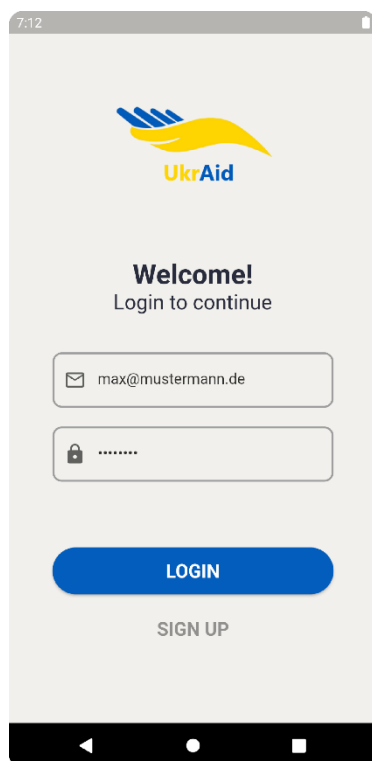


Figure 22 Login Screen

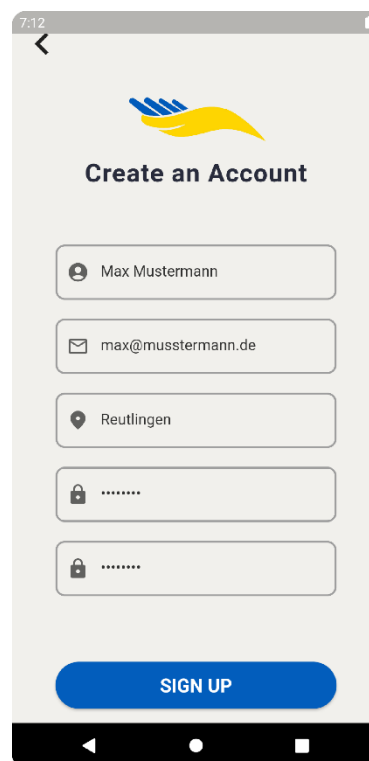


Figure 23 Login Screen

5.1.2 Editing the User Profile

After the user has logged in successfully, he can edit his user profile. Therefore, he can go to the Profile Management View by clicking on the Profile Icon in the NavBar. Here he can click on the Settings Icon and comes to an overview Page, where he either gets to the edit profile information page or can log out. We added this screen to have more options for editing information in the future, such as changing roles or the language.

On the Profile Information Editing Page the user can select a profile picture from the gallery by clicking the “+” and change its name and location. The two text fields must not be empty.

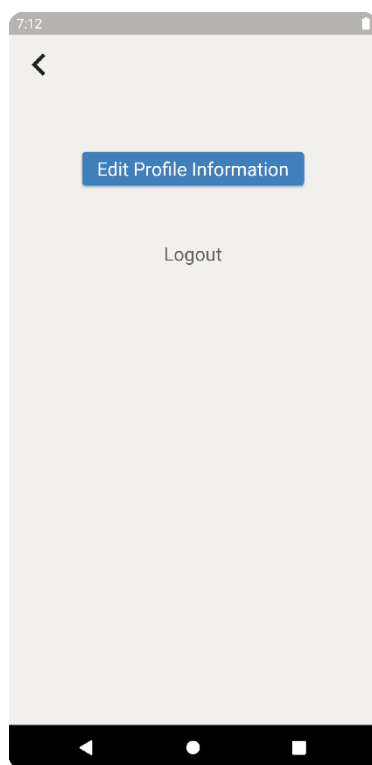


Figure 25 Edit Profile Screen

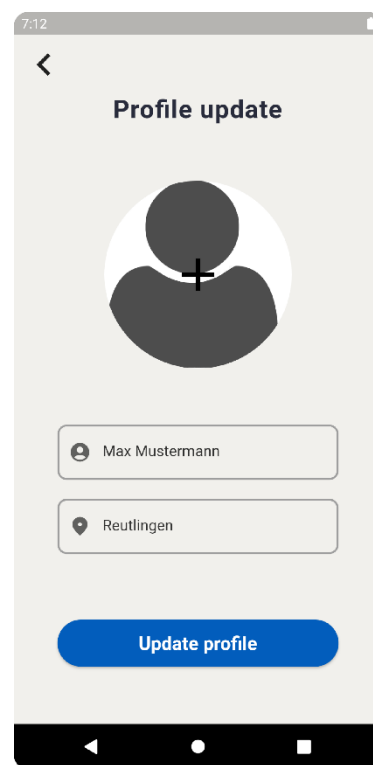


Figure 24 Edit Profile Screen

5.1.3 Create Advertisement (offer type)

The user can click on the + Icon in the Navbar and comes to the Create Advertisement View. Here the user can add images by clicking on the “Add Photo” icon and will then be taken to the mobile phone’s gallery where he can select one or more images. The selected images are then displayed in a scroll bar. Selected images can also be removed by clicking on the “x” icon on the image and more images can also be added. The last element in the scroll bar is the photo icon button. The photos are not necessary, an ad can also be created without pictures. The advertisement has to be given a title with a brief description of the item. This is displayed in the overview of the ads. The titles can there be scanned by the “Search” function.

In addition, the ad has to be assigned to a category of the dropdown and a location where the item is located or can be picked up. A description of the item must also be given. Here the user can describe whether the item can be picked up or delivered.

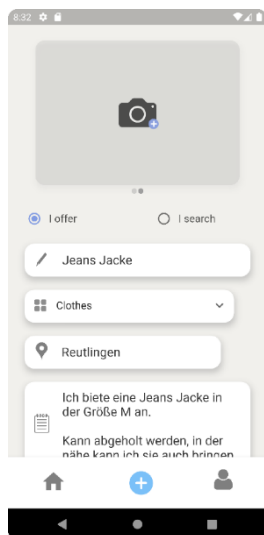


Figure 26 Create Ad Screen

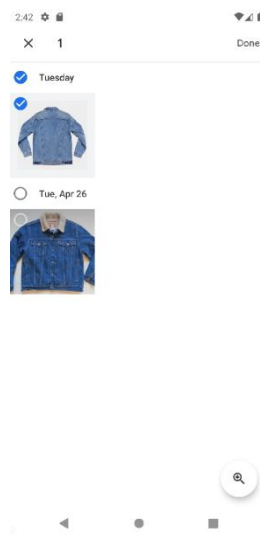


Figure 27 Select picture from device gallery

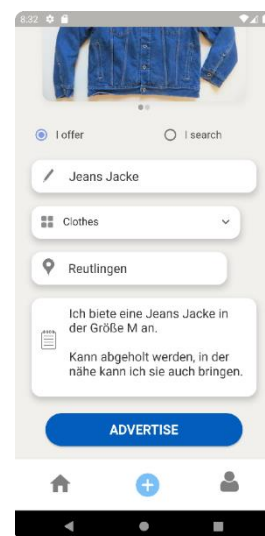


Figure 28 Create Ad Screen with image

5.1.4 Create Advertisement (searches type)

The user can click on the “+” Icon in the Navbar and comes to the Create Advertisement View. There he can select “I search” for the advertisement type by selecting the appropriate radio button. Here the user has the option to fill in the text fields to describe the item he is looking for. The text fields are also mandatory here. The advertisement must be given a title with a brief description of the item the user is looking for. This is displayed in the overview of the ads. In addition, the ad must be assigned to a category from the dropdown and a location where the item is located so the user can pick it up. A description of the item he is looking for must also be made. Here the user also can tell whether he is able to pick up the item.

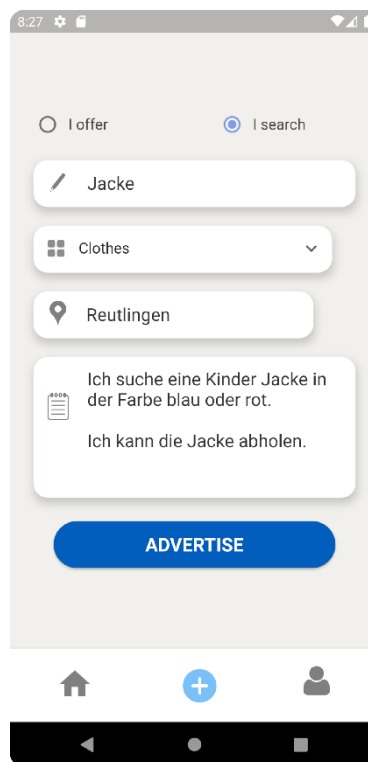
The screenshot shows a mobile application interface for creating an advertisement of the 'search' type. At the top, there are two radio buttons: 'I offer' (unselected) and 'I search' (selected). Below this is a text input field containing the word 'Jacke'. Underneath is a category dropdown menu currently set to 'Clothes'. Below the dropdown is a location input field containing 'Reutlingen'. A larger text area contains two lines of German text: 'Ich suche eine Kinder Jacke in der Farbe blau oder rot.' and 'Ich kann die Jacke abholen.' At the bottom of the form is a prominent blue button labeled 'ADVERTISE'. The app's navigation bar at the very bottom features a home icon, a blue circle with a white plus sign (the active icon), and a user profile icon. The status bar at the top shows the time as 8:27 and various system icons.

Figure 29 Create Ad search

5.1.5 Advertisement Management, Edit and Delete

At the Profile Management View the user can see all active ads he has created. The “given away” button for now has no function, for other versions where advertisements also will have a status (like reserved), this button can be used. By clicking on the “...” Symbol on an advertisement, a context menu appears in which the user can select which action he wants to apply to the selected advertisement. If the user selects the “delete” action, a confirmation popup screen appears and the user can confirm or cancel the deletion. If the user selects “edit”, a screen with all information appears about the ad and the user can edit the text fields and (if the advertisement type is “offer”) select new pictures (similar to the Create Advertisement View).

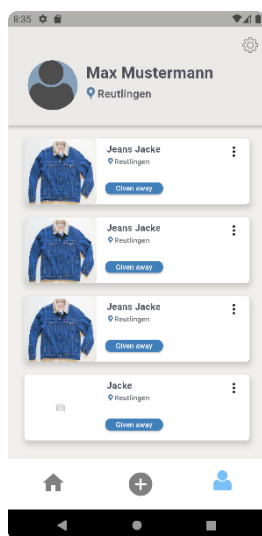


Figure 31 Profile Screen

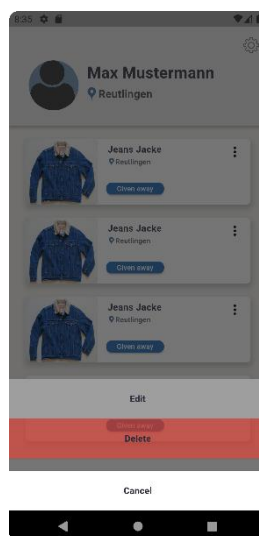


Figure 32 Context menu

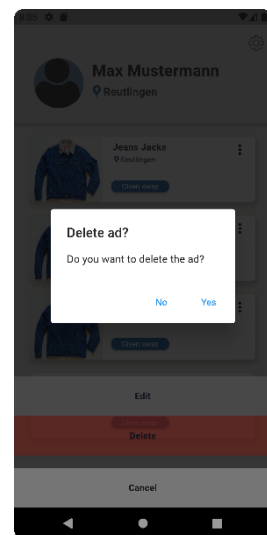


Figure 33 Alert dialog

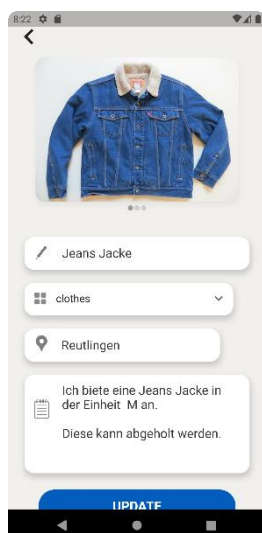


Figure 30 Edit Ad Screen

5.1.6 Search Advertisements

At the Home View of the app (Home Icon) the user can search for specific items. Here the user can find items through various filter options. On the one hand, he can search in a specific category, on the other hand, he can also search in all categories under the option “all”. In addition, he can also determine whether he only wants to see offers or searches, depending on whether he is a refugee or a donor. In the search field the user can search for certain items, i.e., the titles. Unfortunately a translation could not be built in here because the LibreTranslate API did not make it possible to translate in a short request frequency. By clicking on the location, the user can also switch between the current location and all locations. Like this, the user can see other ads if there are none in his city.

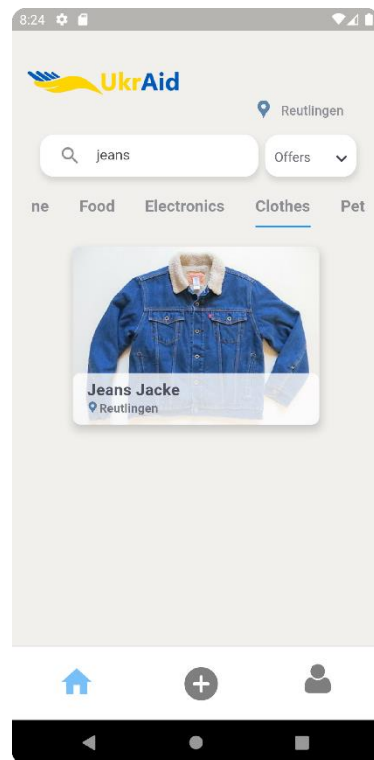


Figure 34 Home Screen search

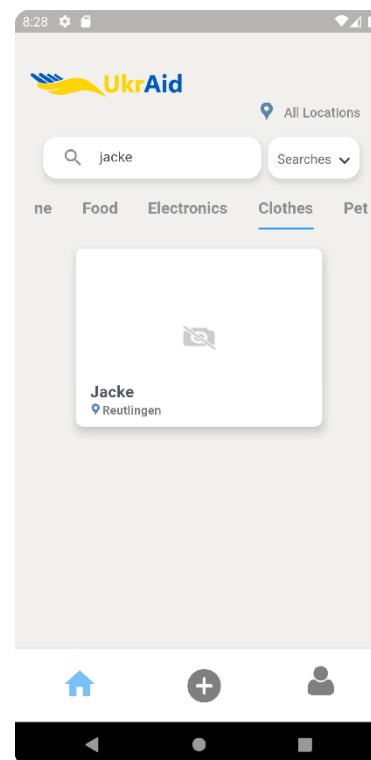


Figure 35 Home screen filtered by search

5.1.7 Home screen filtered by search

By clicking on an advertisement in the Home View, the user can see all details of an add with the pictures, the title, the advertiser name and profile picture and the description. If the user can't understand the ad because the ad was not created in the language, the user can use the translate button to translate the title and description into the language the user used in the app. The translation comes from the translation API. If the user is interested in the ad, he can click on the "contact via Email" button and gets navigated to the phone's email program. There, the "to" field is filled with the email of the advertiser and the subtitle is filled with the title of the ad.

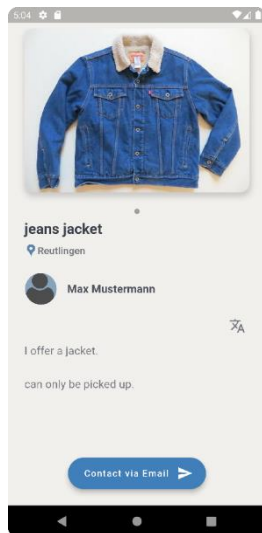


Figure 36 Ad detail screen

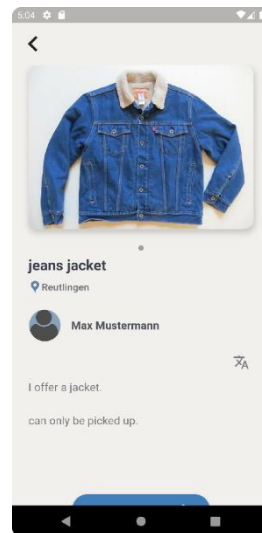


Figure 37 Ad detail screen translated

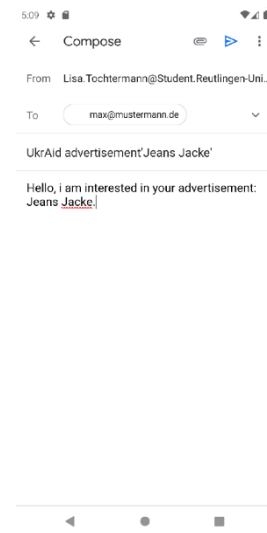


Figure 38 Contact with E-Mail

5.1.8 Example localization

The application can display 3 different languages: German, English and Ukrainian.

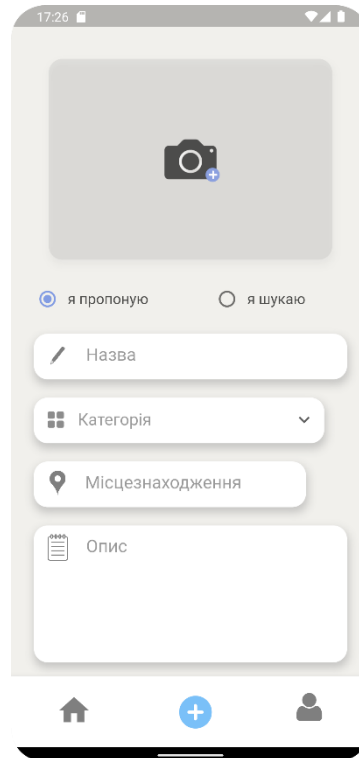


Figure 39 Localization example in Ukrainian

5.2 Degree of completion of the requirements

We managed to implement most of our must have requirements. These must have requirements include F1-F15. F16 as not must have requirement was also successfully implemented, but we ran out of time to implement the may have functional requirements F17-22.

The must have non-functional requirements NF1-NF2 are implemented, while NF3 is only partly done. We ran into a problem with the firebase storage and database which makes it currently not possible to have an appropriate response time when setting images in the storage and referencing them in the database. We had to implement the workaround of artificially waiting for 1 sec to get the correct result in the screens. The may have non-functional requirements NF 4-6 did not make the cut in the application.

The problem with the cache in NF4 was, that this would add another layer of complexity to the application. It is possible that a cache would not reflect the database correctly and thus showing the user wrong information. The implementation of a cache needs to be done correctly, which takes a lot of time that we didn't have. Thus, our application directly gets its data from the Firestore database.

NF5 is hard to comment on, because availability is more of a infrastructure topic. We currently do not host as a backend but use Firebase for this purpose. This means that Firebase is responsible to have their services always available to us. But this also implicitly mean we, as dev using their services, have not much say in this. Our application could not really function if Firebase runs into problems on their side, thus we can't guarantee availability.

NF6 is much like NF5, in the sense that Firebase is also responsible for this aspect. Firebase Authentication for example has sensitive data from our users, therefore have implemented their service to qualify for different ISO norms. For example, the Authentication has ISO 27001, ISO 27017 and ISO 27018 certification. This means the NF6 is partly implemented already by Firebase, but not fully because we have not taken any precautions ourselves.

<https://firebase.google.com/support/privacy>

5.3 Feedback

To get feedback on the application, we already had a meeting with Dr. Gaiduk on 03.06.2022. After a short presentation of the UkrAid application, we were able to get valuable insight on the current situation of Ukrainian refugees in Germany. For example, the current point of contact to get different donations is Telegram or Facebook groups.

Some application specific feedback included:

- Many different categories need to be included
- Ads should be able to be filtered by one or multiple categories
- Text displayed and messages that are being sent to the other party has to be translated/interpreted
- Setting a radius in search function should be possible, so ads with locations that are too far away won't be visible
- It should be possible to see ads in different locations independently from the current location for example different city's
- Required fields should be marked when filling out the ad creation form
- Interpreter meetup function with calendar

Many of those features cannot be implemented in time but could be a good starting point for the second version of the application.

5.4 Challenges

This project was kind of challenging, because mobile development has its unique quirks compared to the normal web development we were used to. For example, different phones can have a different display size that needs to be taken in consideration when making the screens. Another big problem was the lacking emulator in Android Studio that kept crashing often for apparently not reason at all delaying possible progress. Less of a problem but more of a challenge was the usage of a NoSQL database. Up until this point we only used the common relational database model, making this switch in thinking kind of hard. We also had to change the data model in the database often in the process of development, because we realized either a better way to do it or because our current implementation didn't make it possible for some features to be implemented. Sometimes we had to deviate from packages we would have liked to use because the package was not developed further for several months which could have led to problem with new Flutter versions. At the end of the development cycle, we realized we did not reach our goal of features we wanted to achieve, meaning we have many may-haves open. The reason for this is the limited amount of time and the fact that we wanted our base implementation to be as robust as possible.

5.5 Conclusion

We managed to create an application that could potentially help other people in need. But there is still a lot missing to make this a proper application that can fulfil the needs of our target groups. What we have done can be seen as some kind of first version, that needs further development, but is able to at least have the most basic functions we wanted it to have. If UkrAid is destined for further development, a huge refactoring is needed to sort many things out for proper maintainability.

We also gained a lot of practical experience with system architectures and the many different patterns that can be used in this aspect. Having those for the most part implemented meant we had an easier time in the development process because we could for example re use different functions.

But we also learned a lot while developing the application. This was for example our first contact with NoSQL databases and Firebase as whole. Another first was the mobile development with Flutter, as well as the language Dart.

Having this project be a group project also meant we had to use version control and thus properly coordinate with each other. Non proper communication could have led to many merge conflicts, that we managed to avoid for the most part. We regularly held meetings to discuss our development progress, solving problems and discussing further steps.

A unique element in our feedback that other groups most likely did not have is the direct feedback we got from Dr Gaiduk, giving us a unique perspective on the challenges on this topic.

All in all, it was a valuable experience, and gave us a pretty in-depth look in the mobile development domain as well as the problems the people in need have to face and what technology could provide help to them.

6 Table of Figures

Figure 1 Storyboard	4
Figure 2 Persona 1	5
Figure 3 Persona 2	6
Figure 4 Component diagram UkrAid.....	22
Figure 5 Sequence diagram Create User	23
Figure 6 Sequence diagram Create Ad.....	24
Figure 7 Login Screen.....	25
Figure 8 Sign-Up Screen	26
Figure 9 Home Screen.....	27
Figure 10 Ad Detail Screen.....	28
Figure 11 Ad Create Screen with Picture	29
Figure 12 Ad Create Screen	29
Figure 13 Ad Create Screen search.....	30
Figure 14 Profile Screen	31
Figure 15 Layerd pattern UkrAid	32
Figure 16 MVC pattern.....	33
Figure 17 Repository pattern UkrAid.....	34
Figure 18 Repository pattern services UkrAid.....	35
Figure 19 Dependency Injection in general.....	36
Figure 20 Dependency Injection setUp in UkrAid.....	36
Figure 21 Database model UkrAid.....	38
Figure 22 Login Screen.....	40
Figure 23 Login Screen.....	40
Figure 24 Edit Profile Screen	41
Figure 25 Edit Profile Screen	41
Figure 26 Create Ad Screen	42
Figure 27 Select picture from device gallery	42
Figure 28 Create Ad Screen with image	42
Figure 29 Create Ad search	43
Figure 30 Edit Ad Screen	44
Figure 31 Profile Screen	44
Figure 32 Context menu	44
Figure 33 Alert dialog.....	44
Figure 34 Home Screen search	45
Figure 35 Home screen filtered by search	45
Figure 36 Ad detail screen	46
Figure 37 Ad detail screen translated	46
Figure 38 Contact with E-Mail.....	46
Figure 39 Localization example in Ukrainian	47

7 Table Directory

Table 1 Functional requirement – must have – Registration user	13
Table 2 Functional requirement – must have – Login	13
Table 3 Functional requirement – must have – Editing	13
Table 4 Functional requirement – must have – Creation new Ad.....	14
Table 5 Functional requirement – must have – Read ad	14
Table 6 Functional requirement – must have – Edit ad.....	14
Table 7 Functional requirement – must have – Delete ad.....	15
Table 8 Functional requirement – must have – Search ad.....	15
Table 9 Functional requirement – must have – Filter ads by category	15
Table 10 Functional requirement – must have – Pick gallery images	16
Table 11 Functional requirement – must have – Categorize items	16
Table 12 Functional requirement – must have – Filter ads by location	16
Table 13 Functional requirement – must have – Filter ads by type.....	17
Table 14 Functional requirement – must have – Show own ads.....	17
Table 15 Functional requirement – must have – Contact advertiser	17
Table 16 Functional requirement – may have – Translation of ads.....	18
Table 17 Functional requirement – may have – Camera function in app .	18
Table 18 Functional requirement – may have – Geofencing.....	18
Table 19 Functional requirement – may have – Roles for users.....	19
Table 20 Functional requirement – may have – Direct message	19
Table 21 Functional requirement – may have – Real time translation	19
Table 22 Functional requirement – may have – User notification	20
Table 23 Non-functional requirement – must have – OS	20
Table 24 Non-functional requirement – must have – Localization.....	20
Table 25 Non-functional requirement – must have – Response time.....	20
Table 26 Non-functional requirement – may have – Update cache	21
Table 27 Non-functional requirement – may have – Availability	21
Table 28 Non-functional requirement – may have – Security of data.....	21
Table 29 Used dependencies	37

8 Declaration on oath

We hereby declare that we have written the present work on my own, that we have not used any external help other than declared and that all helping material and sources used for the work have been properly named and referenced.

We are aware that an untruthful declaration will be considered as a deceit.

<u>Reutlingen</u>	<u>28.06.2022</u>	<u>Lisa T...</u>
Place,	Date	Signature
		(First and last name)

<u>Reutlingen</u>	<u>28.06.2022</u>	<u>Petrinovic E</u>
Place,	Date	Signature
		(First and last name)