National College of Ireland

BSH in Computing - Evening

Software Development

2020/2021

Denis Novosel

17104718

X17104718@student.ncirl.ie

Handymano

Technical Report

Contents

Executive Summary

1. Introduction

1.1. Background

Throughout last couple of years, I spent large amount of time search for trustworthy professional service providers such as plumbers, electricians and carpenters. Most recent case was when weather started getting colder each day in Ireland and radiators would not heat in my apartment. I did not want to risk getting poor service so I spent some time reading reviews, browsing through Facebook groups and generally browsing the internet for good plumbers.

Throughout my research I found few websites in Ireland that work similarly to my application where user is required to either specify what type of service they require and someone from the company team will reach out back to the user with quotes from different companies/tradespeople or the website requires you to spend significant amount of time describing the service that you need and then post the ad on the public board.

Second type of websites I found were individual websites of tradespeople and business performing the services. All these websites bring initial and maintenance charges and can potentially be inconvenient for the owner to manage content.

Eventually I managed to find the person that has done fantastic job, but I wished there were an easier way. A way where I could see that person's previous work and their verified references – all that only couple of clicks away.

Another feature I wanted was to place an ad where I need some service done such as re-painting the room or lay new set of tiles in my bathroom, but also being able to check the previous project or reviews of the person that offers their service to me.

Lastly, I have decided to call my application "Handymano" because I was not able to find any other application with that name and no domains registered that would conflict with the name of my application.

1.2. Aims

My application targets regular people such as myself who require quality and reliable service and want to receive it without major hassle. I wish to achieve that by creating an application that will allow regular users to find reliable and quality service based on their current location or location they specify as well as the category of the service they require.
Application shows them all the trades and tradespeople that are registered in the system and are in the radius of the user.
Regular users can post job ads for any type of physical work or assistance they require. Any trades or tradespeople registered on the system can offer their service for that job ad.

Professional account types (trades, tradespeople) can customize their profile and essentially create their own portfolio of all the projects they have previously completed. Standard user accounts can provide reviews and feedback of professional accounts.

The application also has basic messaging system through which these two types of users can establish communication and continue with the cooperation.

In my research all the websites I found were overwhelming with the content and each time I visited the website I wanted to close the website.
I can imagine myself as a customer and I want my application to be simple, fluid, easy to use and quick for both types of user. I do not want customers jumping through the hoops for achieving something as simple as finding nearby tradespeople.

1.3. Technology

Android Studio – as my main tool I use Android SDK because I have experience working with Android platform and I own Android device which makes it much easier for me to develop, debug and test this system. In Android Studio I can develop my logic either by using Java or Kotlin. In this case I use Java because I have extensive experience with Java and even though it would be great to learn Kotlin, I believe this project is big and I would not have time to learn Kotlin and deliver the fully completed project.

Firebase – it is cloud platform designed for mobile application development and I the application uses 3 different modules of it. These are:

1. **Authentication** - Firebase offers already completed Authentication module that is easy to integrate with Android Studio and Java code. Besides email and password registration and login it also offers various login methods such as Facebook, Twitter, Google and many more but I will use only email and password authentication.
2. **Firestore** - it is NoSQL cloud database that is ideal for mobile application development. It is easy to store both simple data where best example in my application would be user profile information and complex hierarchical data such as messaging.
3. **Storage** - it is powerful and simple cloud storage. It allows for the storage of images, videos, audio or any other user-generated content. In the case of my application, I will use it for the storage of the image files

Google Cloud – Handymano is using Google Cloud for the utilization of Google Maps API. This API helps with Geofencing to determine registered trades and tradespeople in the radius of the user's current or specified location.

Third-party libraries – To minimize time spent on further development of often boilerplate code, Handymano is making use of the following third-party libraries:

1. **FirebaseUI** – this library offers pre-developed options that allow implementation of registration and login options supported by Firebase. However, I have not leveraged that functionality because I only desired to allow email registration. Instead I used that library for its FirestoreRecyclerAdapter which helped me display data retrieved from the Firebase Firestore through Firestore Query inside of numerous RecyclerViews.

2. **Glide** – it is a library that simplifies image handling within Android environment. Throughout my project I have used Glide for dynamic loading images into the ImageViews as well as for extracting image bitmap and saving local copies of them from the bitmap into the temporary folder on the user's device.

3. **AutoImagesSlider** – this is an open-source library that I have discovered on GitHub which greatly helped me with creating image carousels that are available in Job and Project views.
   Initially I have implemented simple ImageView with motion listener that would detect whether I swiped left or right. Based on that motion detected it would calculate the position and image that needs to be loaded. This was very primitive solution that did not have any animations which made it even less appealing and that is why I decided to rely on already developed solution which is of course free to use and will be referenced.

4. **PrettyTime** – it is an open-source time formatting library that converts standard timestamps into relative human readable timestamps. I have used this library to

format chat activity’s and message menu timestamps that show when the message was last sent.

5. **CircularImageView** – it is an open-source library that I found on GitHub and used it within the application to create round ImageViews instead of default square which I believe now makes application more fluid.

2. System

2.1. Requirements

Functional Requirements

Standard user registrations
Application must allow user to register for the standard account type with email and password.
User is required to provide username, email, password and phone number during the registration.

Professional user registration
Application must allow user to register for the professional account type with email and password.
User is required to provide username, email, phone number, category of services, years of experience, password, location and area willing to travel during the registration.

Navigation drawer
Application must allow user to navigate across the application by utilizing navigation drawer.
Navigation drawer must have different layout for different account types.

View profile
Application must allow user to view their profile information by expanding navigation drawer and clicking on their profile image or username.
Standard account can view professional account’s profile by clicking on it if searched for or if it appears in the list when browsing for services.

Edit profile
Application must allow user to edit their profile information by accessing their profile from navigation drawer.

Delete profile
Application must allow user to delete their account from their profile dashboard.

Log into the application
Application must require user to login. Application must check if user is already logged in.

Log out of the application
Application must allow user to log out.

Create project
Application must allow for project creation at professional account's profile. Project creation must allow for addition of image files, title and description text. After saving project must appear on professional account's profile as the latest project.

View project
Application must allow standard and professional accounts to view projects under professional account's profile

Edit project
Application must allow editing of project at professional account's profile. Project editing must allow user to add or delete image files and text.

Delete project
Application must allow deletion of project from professional's account profile. When project is deleted from the professional account's profile, latest remaining project is added as most recent completed project under professional account's profile.

Browse for tradespeople
Application must allow standard account types to browse for professional accounts and services based on the location specified or the current location of standard account.

Send message
Application must provide functionality to send messages

View message
Application must provide functionality to view message

Delete message
Application must provide functionality to delete message

Create job ad
Application must allow standard account types to create new job ad

View job ad
Application must allow standard and professional account types to view job ad and details

Edit job ad
Application must allow standard account types to edit job ads

Delete job ad
Application must allow standard account types to delete job ads created by their user ID

Respond to job ad
Application must allow professional account types to offer their services for the job

Hire user
Application must allow standard account type to hire professional account for a job ad created

Review and feedback
Application must allow standard account to leave review and feedback of professional account

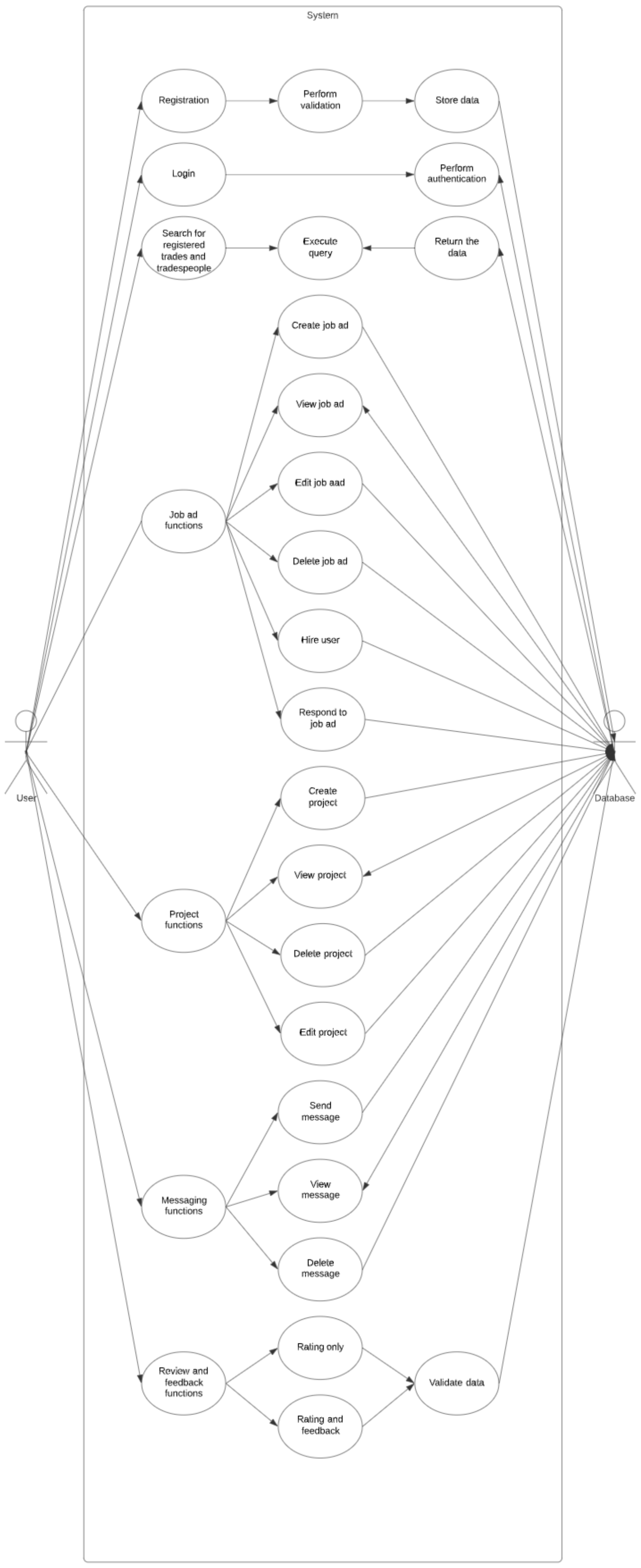


Figure 1 - functional requirements

Requirement 1 – Standard user registration

Description & Priority

User must be able to choose standard account type during the registration. This can be achieved by creating separate button that navigates the user to the activity which creates a flag in the data structure during storing data into the database.

Use Case

Scope

The scope of this use case is to create user account in Firebase data storage with the flag that indicates that user account created is of type "Standard"

Use Case Diagram

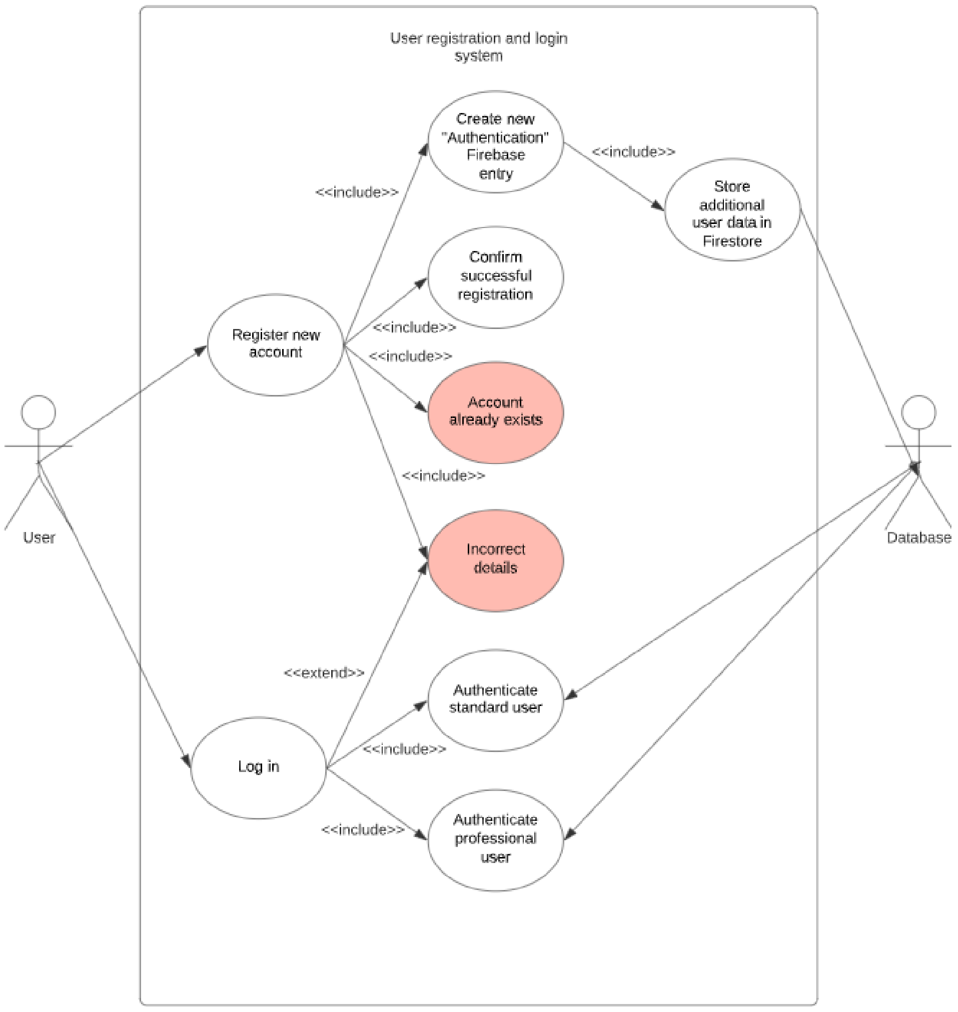

User registration and login system
Create new "Authentication" Firebase entry
<<include>>
Store additional user data in Firestore
<<include>>
Confirm successful registration
Register new account
<<include>>
<<include>>
Account already exists
<<include>>
Incorrect details
User
Database
<<extend>>
Authenticate standard user
<<include>>
Log in
<<include>>
Authenticate professional user

Figure 2 - standard user registration

Flow Description

Precondition

1. Application is installed

2. User is currently not registered
3. User is currently not logged in

Activation

Use case starts when user starts application and clicks on “Register” button

Main flow

1. User starts application
2. User clicks on “Register” button
3. User clicks on button which says “I am looking for a service”
4. User enters required details
5. User clicks on “Register” button
6. Application stores data in Firebase database
7. Data stored in the database for the user contains flag indicating user account is of type “Standard”

Exceptional flow

E1 : Account already exists

1. The system checks if the account is already created
2. If account already exists then it shows an errors and does not proceed with the registration

E2 : User enters incorrect details

1. The system checks if the details entered are in correct format or missing required information
2. If details are incorrect or missing system highlights the input field(s) with incorrect or missing details

Termination

User exits the application

Post condition

The system goes into a wait state

Requirement 2 – Professional user registration

Description & Priority

User must be able to choose professional account type during the registration. This can be achieved by creating separate button that navigates the user to the activity which creates a flag in the data structure during storing data into the database.

Use Case

Scope

The scope of this use case is to create user account in Firebase data storage with the flag that indicates that user account created is of type "Professional"

Use Case Diagram

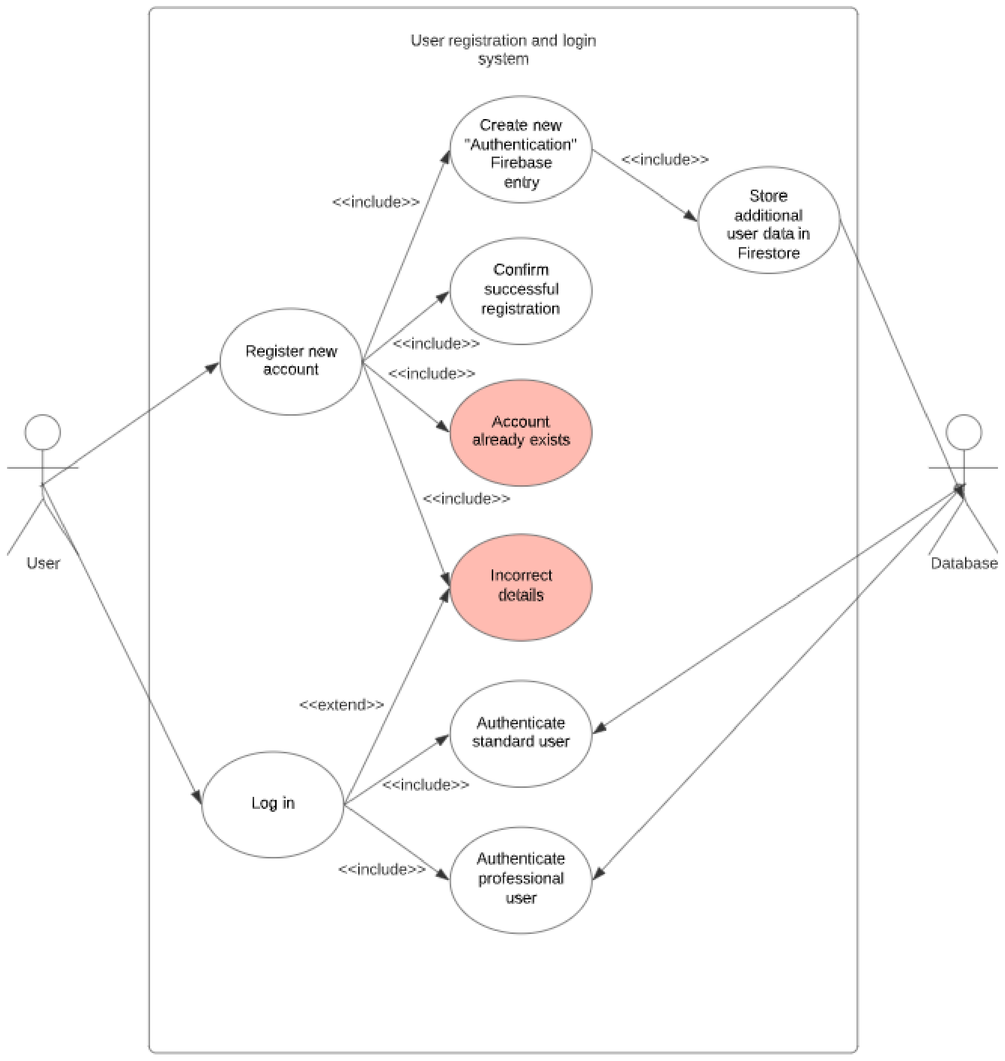

User registration and login system
Create new "Authentication" Firebase entry
<<include>>
Store additional user data in Firestore
<<include>>
Confirm successful registration
Register new account
<<include>>
<<include>>
Account already exists
<<include>>
User
Database
Incorrect details
<<extend>>
Authenticate standard user
Log in
<<include>>
<<include>>
Authenticate professional user

Figure 3 - professional user registration use case

Flow Description

Precondition

1. Application is installed
2. User is currently not registered
3. User is currently not logged in

Activation

Use case starts when user starts application and clicks on "Register" button

Main flow

1. User starts application
2. User clicks on "Register" button

3. User clicks on button which says "I am offering a service"
4. User enters required details
5. User clicks on "Register" button
6. Application stores data in Firebase database
7. Data stored in the database for the user contains flag indicating user account is of type "Professional"

Exceptional flow

E1 : Account already exists

1. The system checks if the account is already created
2. If account already exists then it shows an errors and does not proceed with the registration

E2 : User enters incorrect details

1. The system checks if the details entered are in correct format or missing required information
2. If details are incorrect or missing system highlights the input field(s) with incorrect or missing details

Termination

User exits the application

Post condition

The system goes into a wait state

Requirement 3 – Navigation drawer

Description & Priority

Application must contain the navigation drawer that will contain different navigation items based on the currently logged in account type. Navigation drawer must not be visible until user presses on the button which expands the navigation drawer.
Navigation drawer must be collapsible if user presses on the button which collapses the navigation drawer or presses area of the screen outside of the navigation drawer boundaries.

Use Case

Scope

The scope of this use case is to create method of navigation throughout different application screens

Use Case Diagram

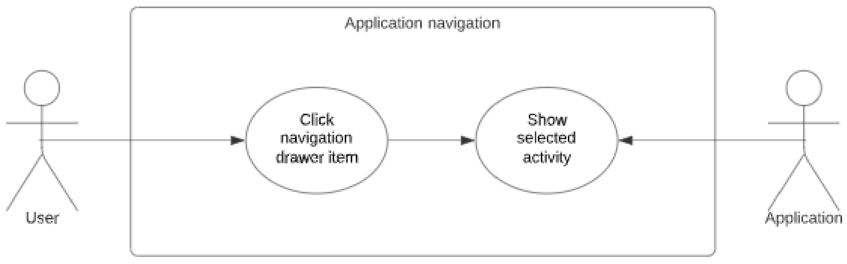


Figure 4 - navigation drawer use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged in

Activation

Use case starts when user starts application and presses the button for expansion of navigation drawer

Main flow

1. User presses button to expand the navigation drawer
2. User presses one of the buttons on the navigation drawer for navigation to desired activity
3. User is transferred to desired activity
4. Navigation drawer collapses

Termination

User exits the application, presses area of the screen outside of the navigation drawer boundaries or presses the button to collapse navigation drawer

Post condition

The system goes into a wait state

Requirement 4 – View profile

Description & Priority

Application must allow user to visit their profile and see information stored under their profile.

Application must also allow Standard users to view profile of Professional account types.

Use Case

Scope

The scope of this use case is to create functionality for users to view profiles by retrieving data from the database.

Use Case Diagram

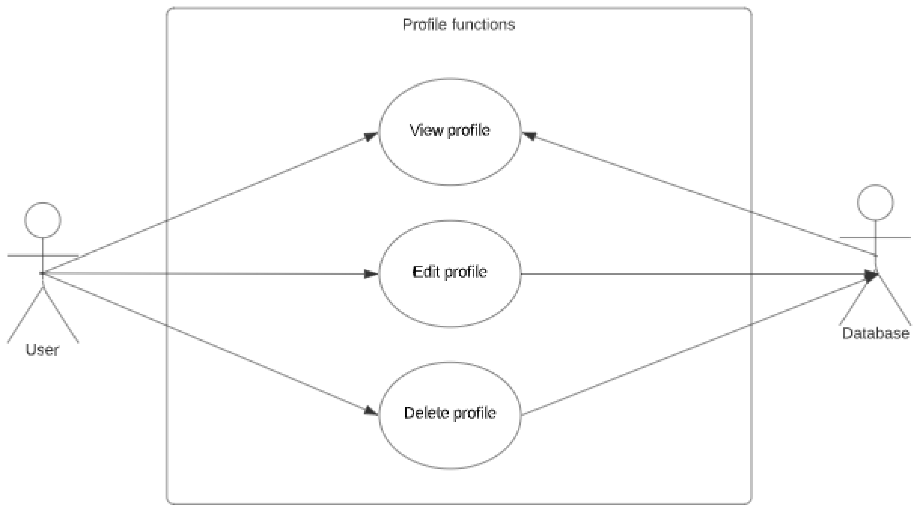


Figure 5 - view profile use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged in.

Activation

Use case starts when user starts application, expands navigation drawer and clicks on their profile.

Main flow

1. User presses the button to expand navigation drawer
2. User presses image of their profile or their username
3. User is transferred to their profile activity

Alternate flow

1. User clicks on the username of the professional account
2. User is transferred to the professional account's profile activity

Termination

User exits the application or user navigates to the previous activity

Post condition

The system goes into a wait state

Requirement 5 – Edit profile

Description & Priority

Application must allow both account types to edit information stored under their profile.

Use Case

Scope

The scope of this use case is to create functionality for users to edit their profiles by accessing their profile, modifying the data under profile and pressing the button to save their changes. User will be prompted to confirm if they wish to save the changes and if they confirm then update request is sent to the database.

Use Case Diagram

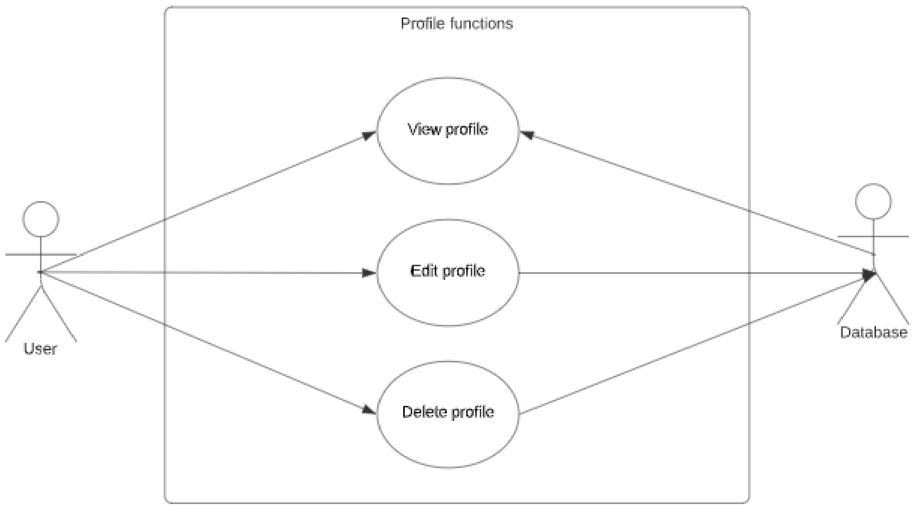


Figure 6 - edit profile use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged in.

Activation

Use case starts when user starts application, expands navigation drawer and clicks on their profile.

Main flow

1. User presses the button to expand navigation drawer
2. User presses image of their profile or their username
3. User is transferred to their profile activity
4. User presses on the editing icon

5. User is presented with the fields they can change
6. User clicks on button to save the changes

Exceptional flow

E1 : User enters incorrect details

1. The system checks if the details entered are in correct format or missing required information
2. If details are incorrect or missing system highlights the input field(s) with incorrect or missing details

Termination

User exits the application or cancels the editing

Post condition

The system goes into a wait state

Requirement 6 – Delete profile

Description & Priority

Application must allow users to delete their profiles.

Use Case

Scope

The scope of this use case is to explain functionality of deleting user profile by accessing it and selecting the option to delete profile. After the option is confirmed, user profile data is deleted from the database.

Use Case Diagram

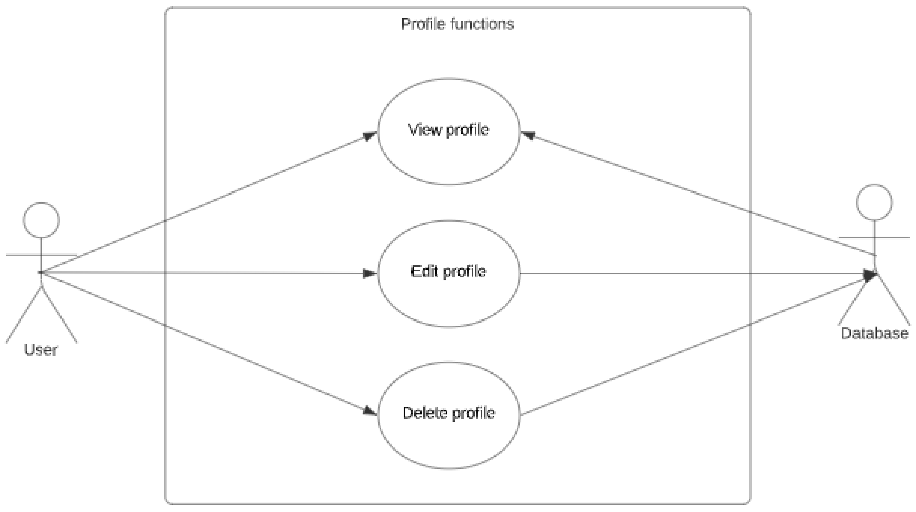


Figure 7 - delete profile use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged in

Activation

Use case starts when user starts application, expands navigation drawer and clicks on their profile.

Main flow

1. User presses the button to expand navigation drawer
2. User presses image of their profile or their username
3. User is transferred to their profile activity
4. User presses on the advanced options
5. User presses on button to delete profile
6. User is prompted to confirm their decision
7. Profile is deleted
8. User is logged out of the application

Termination

User exits the application or cancels the deletion

Post condition

The system goes into a wait state

Requirement 7 – Log into the application

Description & Priority

Application must have ability to distinguish whether user is registered and authenticate them based on whether their credentials are valid.

Use Case

Scope

The scope of this use case is to allow authenticated users further access to application functionality

Use Case Diagram

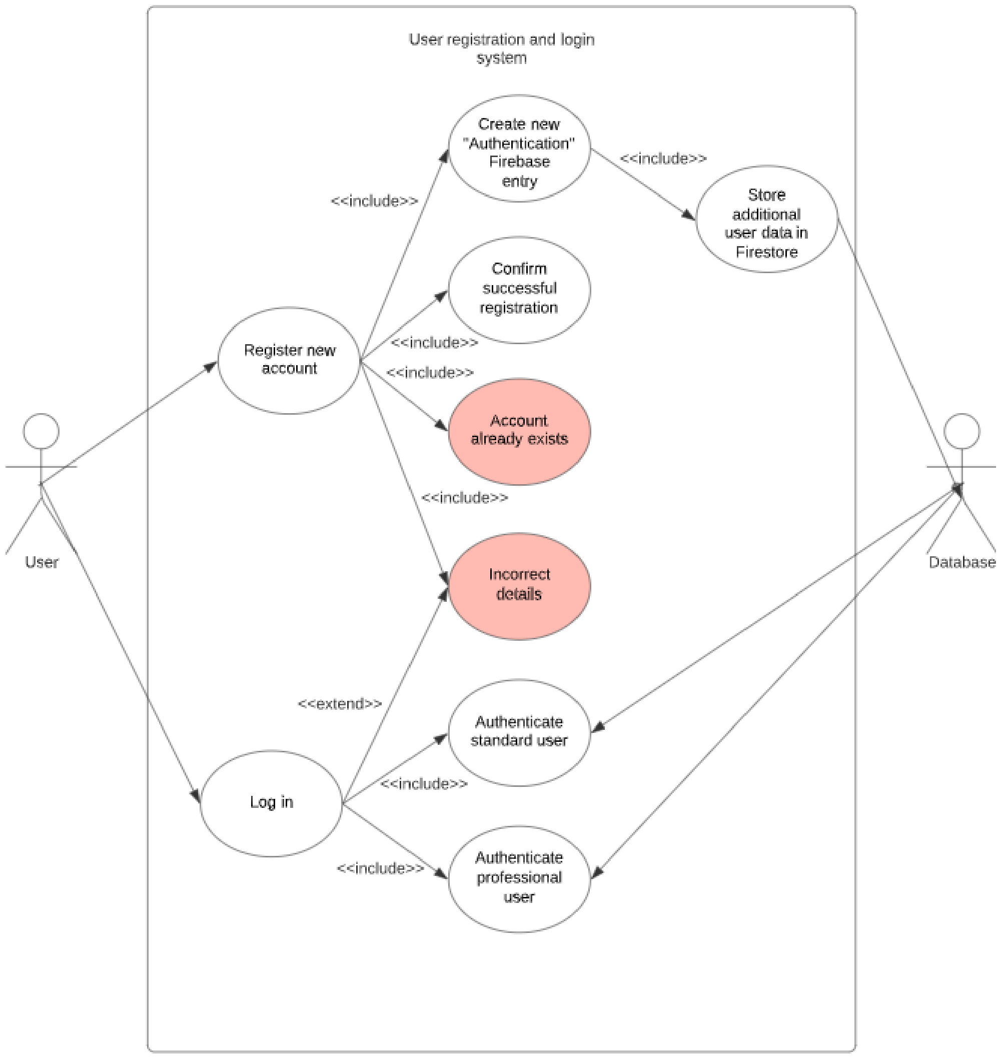

User registration and login system
Create new "Authentication" Firebase entry
<<include>>
Store additional user data in Firestore
<<include>>
Confirm successful registration
Register new account
<<include>>
<<include>>
Account already exists
User
Database
<<include>>
Incorrect details
<<extend>>
Authenticate standard user
<<include>>
Log in
<<include>>
Authenticate professional user

Figure 8 - login use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged out

Activation

Use case starts when user starts application

Main flow

1. User starts the application
2. User enters email and password
3. User clicks on “Login” button
4. User is transferred to next activity

Exceptional flow

E1 : User enters incorrect details

1. The system checks if the details entered are correct format or missing required information
2. If details are incorrect or missing, system highlights the input field(s) with incorrect or missing details

Termination

User exits the application

Post condition

The system goes into a wait state

Requirement 8– Log out of the application

Description & Priority

Application must have the ability that allows user to log out of application. This should set the flag that user is logged out and user is required to log into the application on next launch

Use Case

Scope

The scope of this use case is to set user as logged out and prevent any further access to application functionality other than being logged into the application.

Use Case Diagram

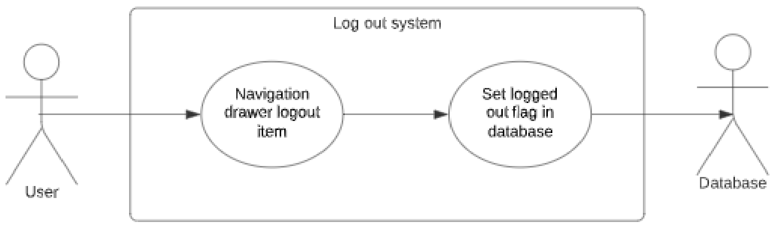

Log out system
Navigation drawer logout item
Set logged out flag in database
User
Database

Figure 9 - logout use case

Flow Description

Precondition

Application is installed, user is registered and user is logged into the application.

Activation

Use case starts when user starts application

Main flow

1. User starts the application
2. User clicks on button to expand navigation drawer

3. User clicks on “Log out” button on navigation drawer
4. User is prompted to confirm their decision
5. User is logged out of application
6. User is transferred to the login activity

Termination

User exits the application or user cancels the log out

Post condition

The system goes into a wait state

Requirement 9 – Create project

Description & Priority

Application must have the ability that allows professional account types to create project that can be visible under professional account’s profile

Use Case

Scope

The scope of this use case is to save project details and images under user’s collection in database.

Use Case Diagram

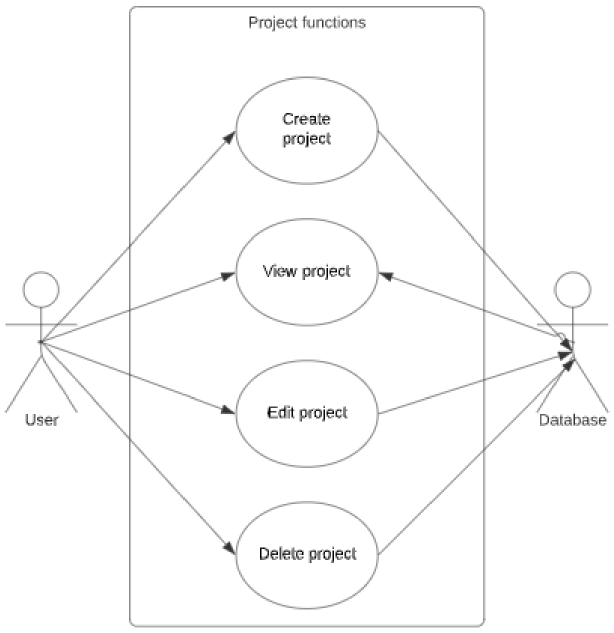


Figure 10 - create project use case

Flow Description

Precondition

1. Application is installed
2. User is registered

3. User is logged into the application
4. User's account type is "Professional User"

Activation

Use case starts when user clicks on navigation drawer item to access projects

Main flow

1. User clicks on "Projects" navigation drawer item
2. User is transferred to "Projects" activity
3. User clicks on "Create New" button
4. User enters required details regarding the project and attaches image files
5. User clicks on "Save" button to store the changes in the database
6. Project is added as the most recent project under Professional account's profile

Termination

T1: User exits the application

T2: User cancels the creation of new project

Post condition

The system goes into a wait state

Requirement 10 – View project

Description & Priority

Application must have the ability that allows user to view project and details stored for that project

Use Case

Scope

The scope of this use case is to create functionality that will retrieve data from database for specific project user selects

Use Case Diagram

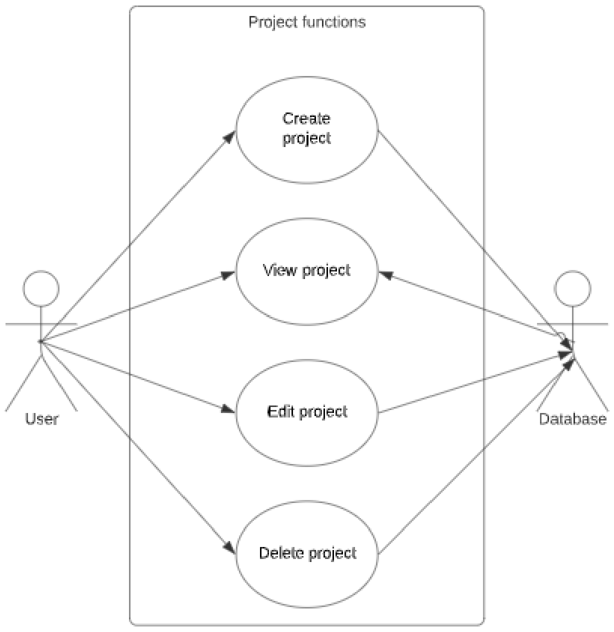


Figure 11 - view project use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. Project exists

Activation

Use case starts when user clicks on specific project

Main flow

1. User clicks on the button which references the specific project such as the project listed on professional account's profile
2. User is transferred to the activity containing project details and images

Termination

T1: User exits the application

T2: User navigates back to previous activity

Post condition

The system goes into a wait state

Requirement 11 – Edit project

Description & Priority

Application must have the ability that allows user to view project and details stored for that project under their profile

Use Case

Scope

The scope of this use case is to allow professional user to edit the project they access under their profile and save the changes into the database.

Use Case Diagram

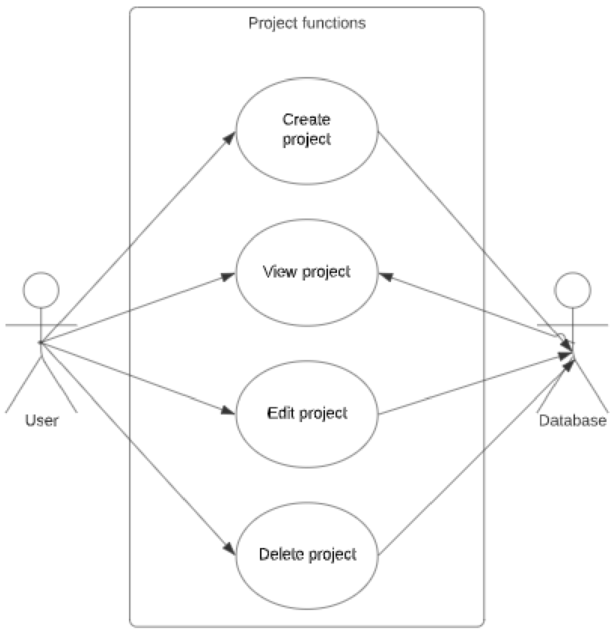


Figure 12 - edit project use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Professional User"
5. Project exists

Activation

Use case starts when user clicks on navigation drawer item to access projects

Main flow

1. User clicks on “Projects” navigation drawer item
2. User is transferred to “Projects” activity
3. User clicks on one of the existing projects
4. User clicks on “Edit project” button and makes desired changes
5. User clicks on “Save” button to store the changes in the database

Termination

T1: User exits the application

T2: User cancels the changes of project

Post condition

The system goes into a wait state

Requirement 12 – Delete project

Description & Priority

Application must have the ability to allow professional users to delete projects under their profile

Use Case

Scope

The scope of this use case is to create functionality that will allow professional users to access their projects and delete them. Professional user will be prompted to confirm their decision and if application receives confirmation than project will be deleted from the database.

Use Case Diagram

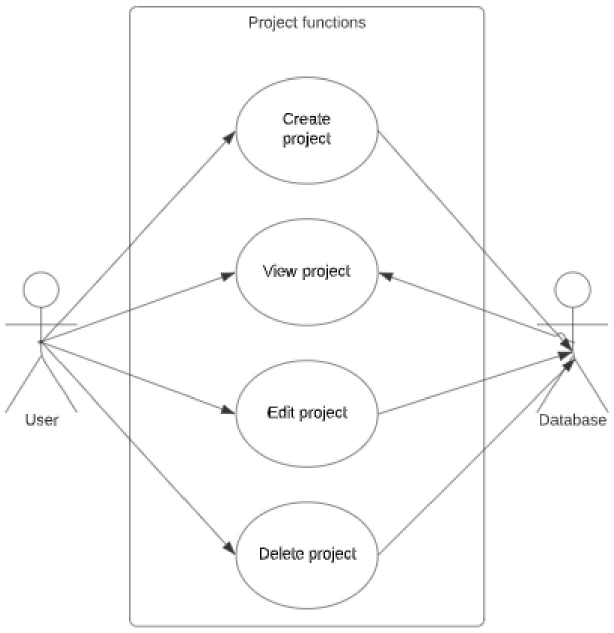


Figure 13 - delete project use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Professional User"
5. Project exists

Activation

Use case starts when user clicks on navigation drawer item to access projects

Main flow

1. User clicks on "Projects" navigation drawer item
2. User is transferred to "Projects" activity
3. User clicks on one of the existing projects
4. User clicks on "Delete project" button
5. Application prompts user to confirm their decision
6. Project is deleted from the database and user is transferred to "Projects" activity
7. If most recent project is deleted, next most recent project appears as most recent project under professional account's profile

Termination

T1: User exits the application

T2: User cancels the changes of project

Post condition

The system goes into a wait state

Requirement 13 – Browse for tradespeople

Description & Priority

Application must have the ability that will allow standard users to browse for the registered tradespeople and trades. Results are to be returned from the database in the form of scrollable list.

Use Case

Scope

The scope of this use case is to create functionality that will retrieve users from the database based on the query which will entail number of different parameters.

Use Case Diagram

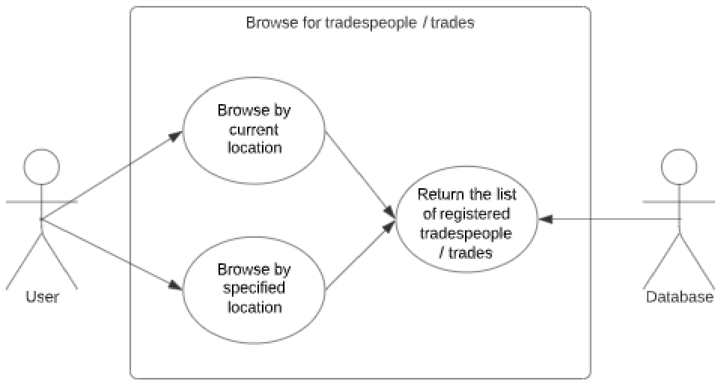

Browse for tradespeople / trades
Browse by current location
Browse by specified location
Return the list of registered tradespeople / trades
User
Database

Figure 14 - browse for tradespeople use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Standard User"

Activation

Use case starts when user clicks on button to browse for tradespeople

Main flow

1. User selects category of services to browse for
2. User leaves default option "Current location" as the location
3. User clicks on search button
4. Application shows list of trades and tradespeople based on the users location

Alternate flow

1. User selects category of services to browse for
2. User enters location to browse at
3. User clicks on search button
4. Application shows list of trades and tradespeople based on the users location

Termination

T1: User exits the application

Post condition

The system goes into a wait state

Requirement 14 – Send message

Description & Priority

Application must have the ability that will allow users to send messages and replies to the other users.

Use Case

Scope

The scope of this use case is to create functionality that will allow communication between users. Messages will be stored as the collection in the database.

Use Case Diagram

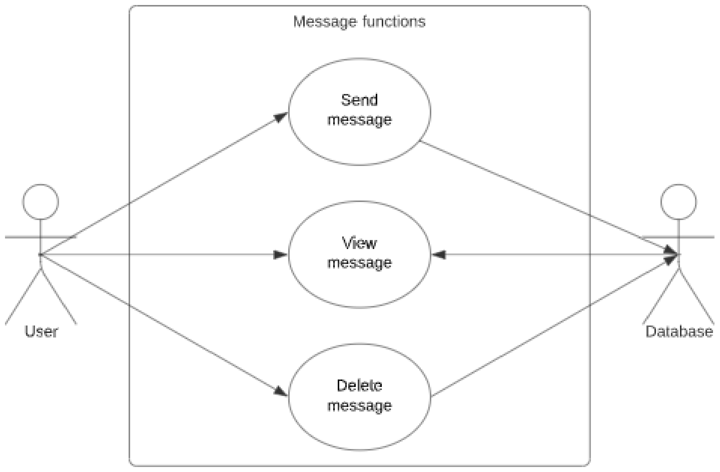


Figure 15 - send message use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application

Activation

Use case starts when user clicks on button to send message to user

Main flow

1. Standard user clicks on “Message” button on professional’s account profile
2. Standard user is presented with the message form
3. Standard user enters desired text into message form
4. Standard user clicks on “Send” button
5. Message is sent to professional user
6. Standard user is transferred to “Messages” activity

Alternate flow

1. Professional user opens message

2. Professional user enters their message
3. Professional user clicks on “Send” button
4. Message is sent to standard user
5. Professional user is transferred to “Messages” activity

Exceptional flow
E1: User does not enter text when sending message
1. User does not enter any text into message form
2. User clicks on “Send” button
3. Message is not sent
4. Application highlights field with error

Termination

T1: User exits the application

T2: User cancels message

Post condition

The system goes into a wait state

Requirement 15 – View message

Description & Priority

Application must have the ability that will allow user to view message that was sent to them.

Use Case

Scope

The scope of this use case is to create functionality that will allow communication between users.

Use Case Diagram

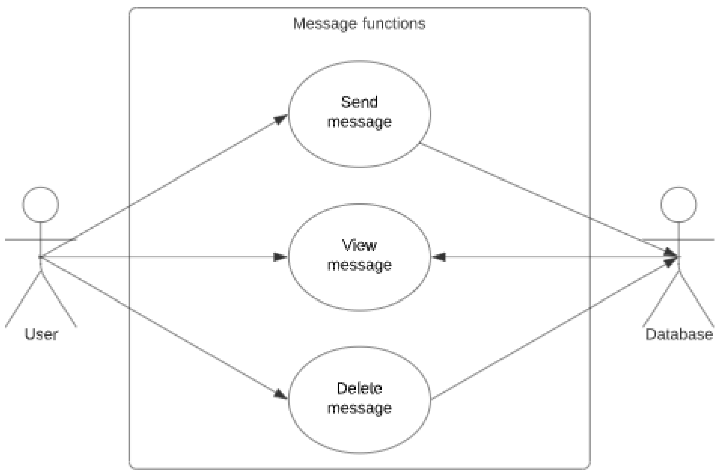


Figure 16 - view message use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. Message exists

Activation

Use case starts when user clicks on button to open messages activity

Main flow

1. User clicks on “Messages” button on professional’s account profile
2. User clicks on message they wish to view
3. Application shows message contents to the user

Termination

T1: User exits the application

T2: User navigates to previous activity

Post condition

The system goes into a wait state

Requirement 16 – Delete message

Description & Priority

Application must have the ability that will allow user to delete the message that was sent to them.

Use Case

Scope

The scope of this use case is to create functionality that will delete the message from the database.

Use Case Diagram

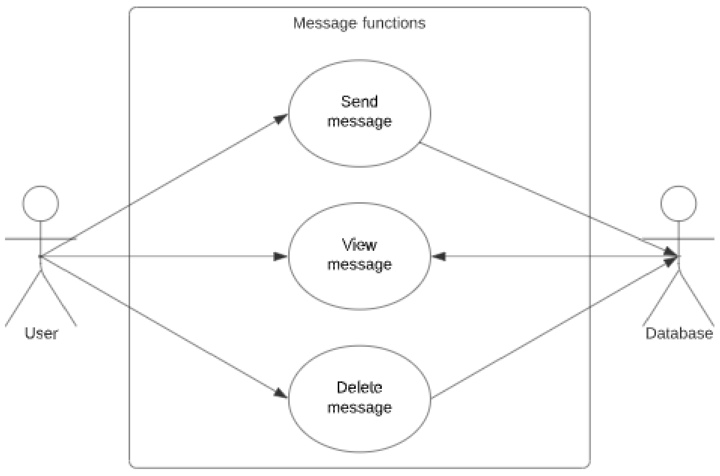


Figure 17 - delete message use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. Message exists

Activation

Use case starts when user clicks on button to open messages activity

Main flow

1. User clicks on “Messages” button on professional’s account profile
2. User clicks on message they wish to view
3. Application shows message contents to the user
4. User clicks on “Delete message” button
5. Application prompts user for confirmation to delete the message
6. User confirms the choice to delete the message
7. Message is deleted from database
8. User is transferred to previous activity

Termination

T1: User exits the application

T2: User navigates to previous activity

T3: User does not delete the message

Post condition

The system goes into a wait state

Requirement 17 – Create job ad

Description & Priority

Application must have the ability that will allow standard user to create job ads. Structure of job ads will consist of text and images. Upon creation of job ad it will be stored in the database.

Use Case

Scope

The scope of this use case is to create functionality that will store job ad data into the database.

Use Case Diagram

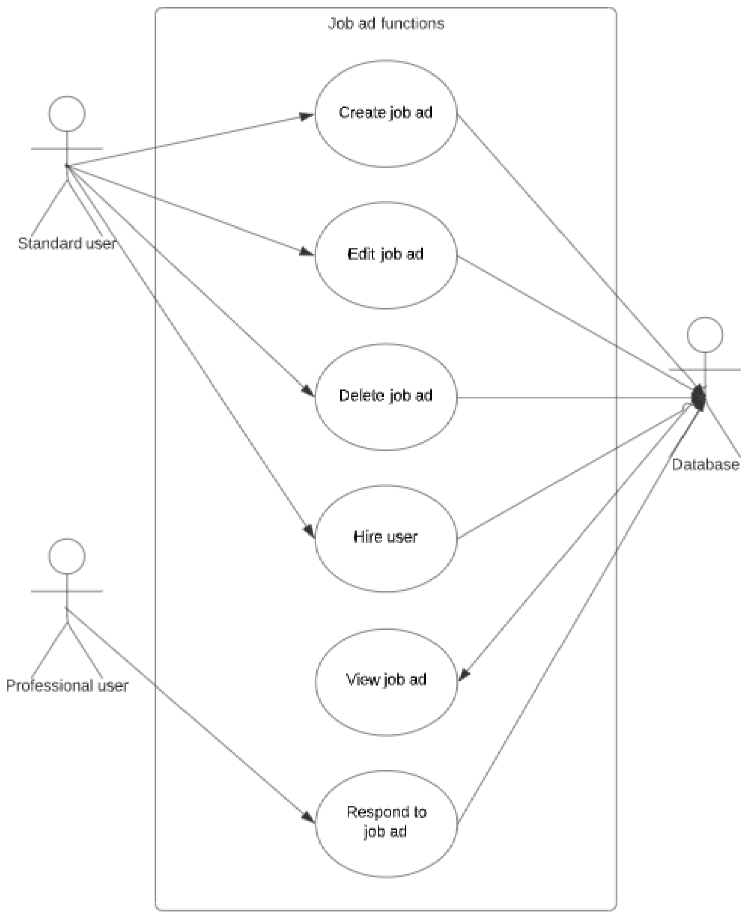


Figure 18 - create job ad use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application

4. User's account type is "Standard User"

Activation

Use case starts when user clicks on navigation drawer item to access job ads

Main flow

1. User clicks on "Job ads" navigation drawer item
2. User is transferred to "Job ads" activity
3. User clicks on "Create New" button
4. User enters required details regarding the job and attaches image files (optional)
5. User clicks on "Save" button to store the changes in the database
6. Job is added to the job board as the most recent job

Termination

T1: User exits the application

T2: User cancels the creation of new job

Post condition

The system goes into a wait state

Requirement 18 – View job ad

Description & Priority

Application must have the ability to allow users to view job ads. Job ads will consist of text and images.

Use Case

Scope

The scope of this use case is to create functionality that will retrieve job ad data from the database and show it to the user.

Use Case Diagram

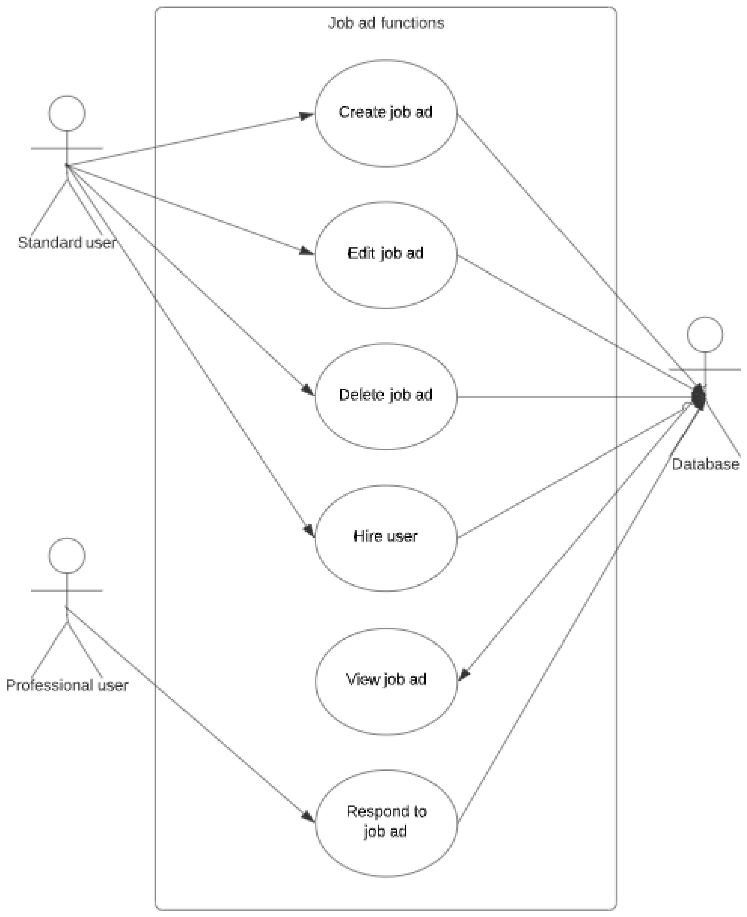


Figure 19 - view job use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. Job ad exists

Activation

Use case starts when user clicks on specific job ad

Main flow

1. User clicks on the button which references the specific job ad
2. User is transferred to the activity containing job ad details and images

Termination

T1: User exits the application

T2: User navigates back to previous activity

Post condition

The system goes into a wait state

Requirement 19 – Edit job ad

Description & Priority

Application must have the ability that will allow author of the job ad to edit job ad and save the changes to the database. Before saving the data user will be prompted to confirm their decision. If decision is positive then data will be updated.

Use Case

Scope

The scope of this use case is to create functionality that will allow user to edit the job ads they created.

Use Case Diagram

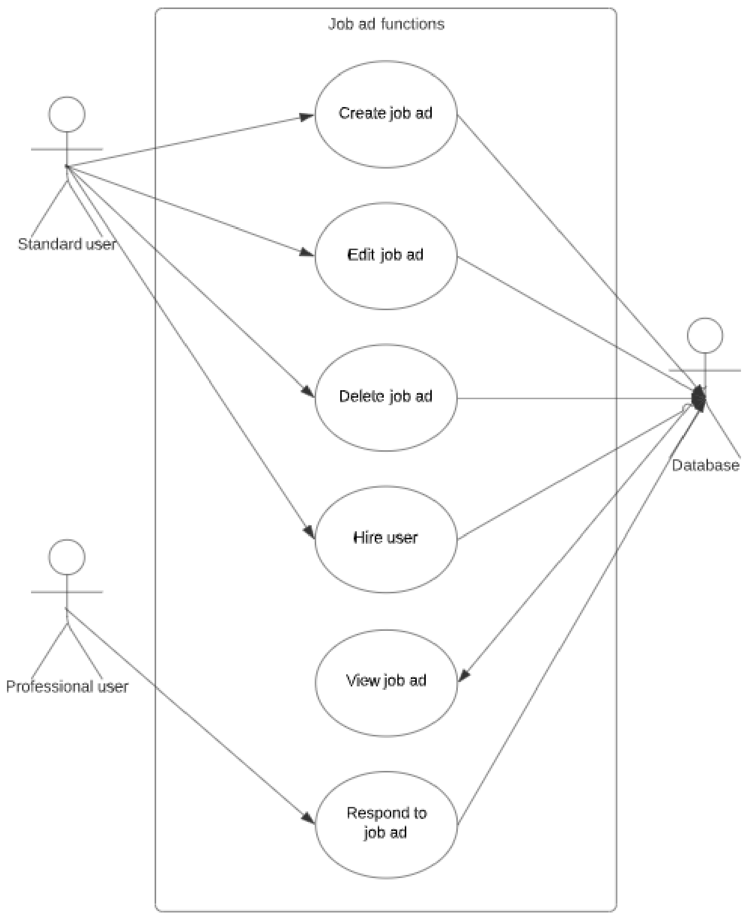


Figure 20 - edit job ad use case

Flow Description

Precondition

1. Application is installed
2. User is registered

3. User is logged into the application
4. User's account type is "Standard User"
5. Job ad exists

Activation

Use case starts when user clicks on navigation drawer item to access job ads

Main flow

1. User clicks on "Job ads" navigation drawer item
2. User is transferred to "Job ads" activity
3. User clicks on one of the existing job ads
4. User clicks on "Edit ad" button and makes desired changes
5. User clicks on "Save" button to store the changes in the database

Termination

T1: User exits the application

T2: User cancels the changes of ad

Post condition

The system goes into a wait state

Requirement 20 – Delete job ad

Description & Priority

Application must have the ability that will allow job ad authors to delete job ads they created. User will be prompted to confirm their decision. If the decision is positive job ad will be deleted from the database.

Use Case

Scope

The scope of this use case is to create functionality that will delete job ad from the database.

Use Case Diagram

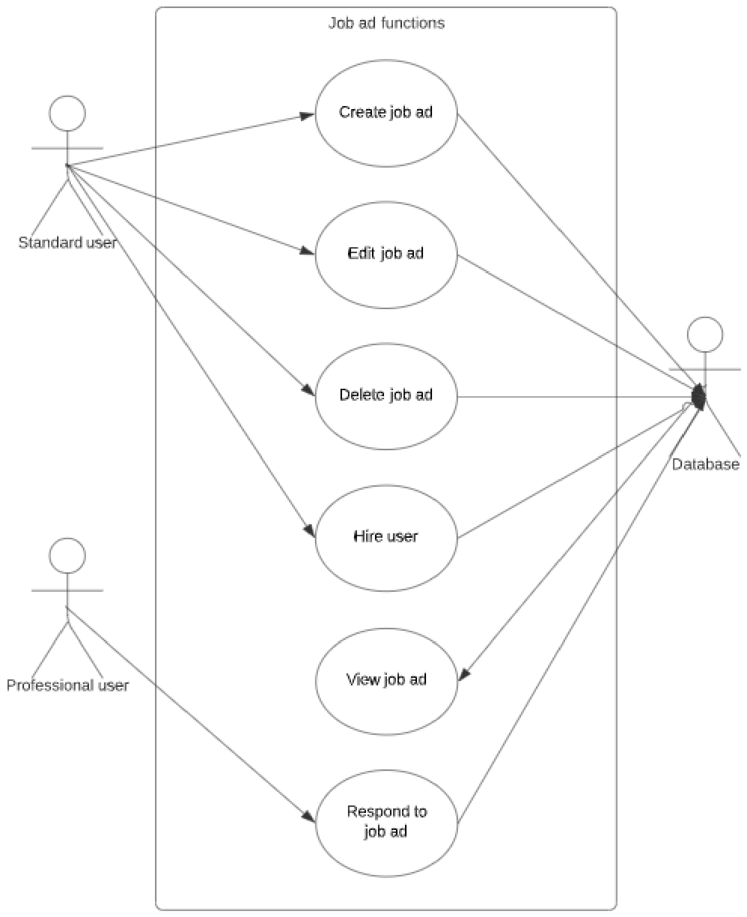


Figure 21 - delete job ad use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Standard User"
5. Job ad exists

Activation

Use case starts when user clicks on navigation drawer item to access job ads

Main flow

1. User clicks on "Job ads" navigation drawer item
2. User is transferred to "Job ads" activity
3. User clicks on one of the existing job ads
4. User clicks on "Delete job ad" button
5. Application prompts user to confirm their decision
6. Job ad is deleted from the database and user is transferred to "Job ads" activity

Termination

T1: User exits the application

T2: User cancels the deletion of job ad

Post condition

The system goes into a wait state

Requirement 21 – Respond to job ad

Description & Priority

Application must have the ability that will allow professional users to respond to the job ad. When professional user responds to the job ad, a message is sent to job ad author with the title of the job ad and message that professional user composes. Users can continue messaging from this original message.

Use Case

Scope

The scope of this use case is to create functionality that will allow professional users send messages from the job ads to the job ad authors.

Use Case Diagram

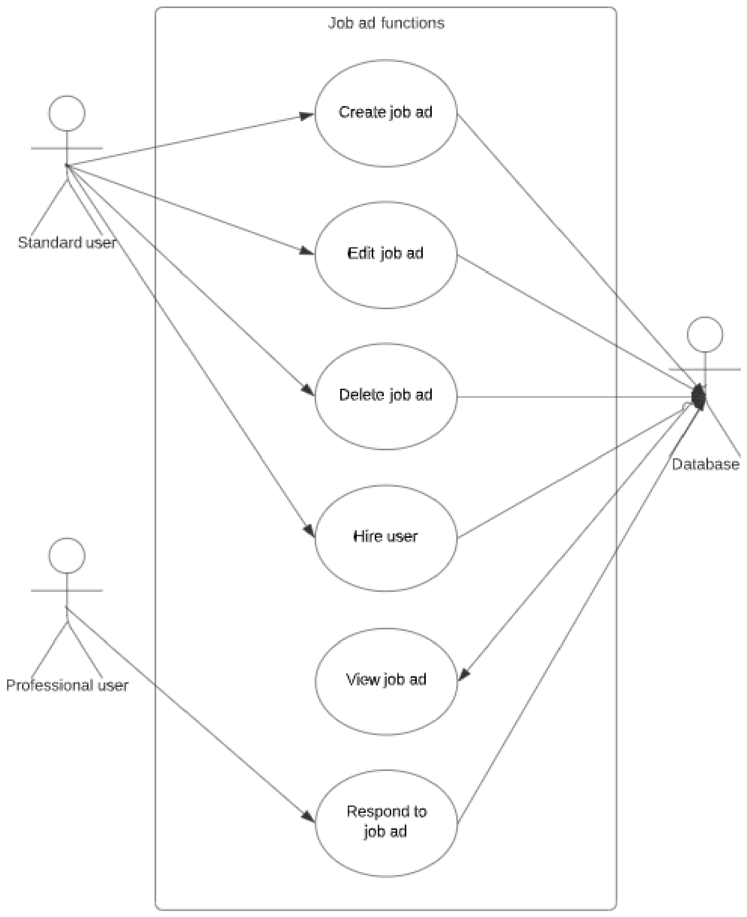


Figure 22 - respond to job ad use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Professional User"
5. Job ad exists

Activation

Use case starts when user clicks on navigation drawer item to access job ads

Main flow

1. User clicks on "Job ads" navigation drawer item
2. User is transferred to "Job ads" activity
3. User clicks on one of the existing job ads
4. User clicks on "Message advertiser" button
5. Application opens messaging form
6. User enters details with their offer for job ad
7. User clicks on "Send message" icon

8. Application sends offer message to job ad creator

Termination

T1: User exits the application

T2: User cancels sending offer message

Post condition

The system goes into a wait state

Requirement 22 – Review and feedback

Description & Priority

Application must have the ability that will allow standard users to leave their feedback, review and/or rating of the professional users. Feedback shall be visible to the other users.

Use Case

Scope

The scope of this use case is to create functionality that will allow standard users to rate professional users and store rating data into the database for the professional user's ID.

Use Case Diagram

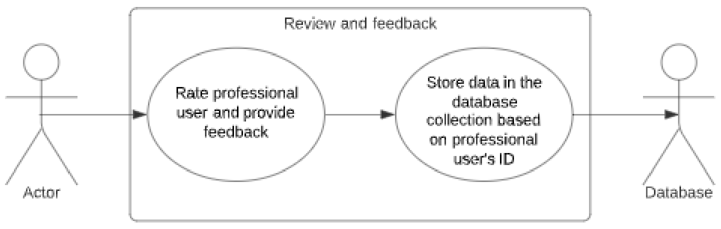

Review and feedback
Actor
Rate professional user and provide feedback
Store data in the database collection based on professional user's ID
Database

Figure 23 - review and feedback use case

Flow Description

Precondition

1. Application is installed
2. User is registered
3. User is logged into the application
4. User's account type is "Standard User"

Activation

Use case starts when user clicks on navigation drawer item to access job ads

Main flow

1. User views profile of tradesman or trade
2. User selects rating (1-10)
3. User enters feedback (Optional)
4. User clicks on “Post review”
5. Data is stored in the database
6. Professional accounts rating is updated
7. User is returned to previous activity

Termination

T1: User exits the application

T2: User navigates to previous activity

Post condition

The system goes into a wait state

Non-functional requirements

Requirement 1 – Availability

Application must be always available and this is achieved by publishing the application onto Google Play store where users can always download the application.
Due to all of the application functionalities utilizing Google Cloud services (Maps, Firebase), application availability and uptime is guaranteed.

Requirement 2 – Security

Interaction with the application must be performed in the secure manner. The only authentication method that is available now is through Firebase’s email and password authentication module. This interface provides secure model and the application does not have access to user’s details except for the email.

Requirement 3 – Responsiveness

Application must be able to fetch and process data quickly without noticeable delays. Where the delays are necessary such as during the asynchronous calls, application must let the user know of the status.

Requirement 4 – Error handling

Application must notify the user of the about any issues with their request or application errors in a user understandable format instead of printing stack trace which regular users would not be able to understand.

2.2. Design & Architecture

Firebase is main platform where the application is storing data and receiving it back. Google Cloud is platform where Google Maps API will be utilized for further application functions.

Below is the low-level overview of the application and its interaction with the Google services.

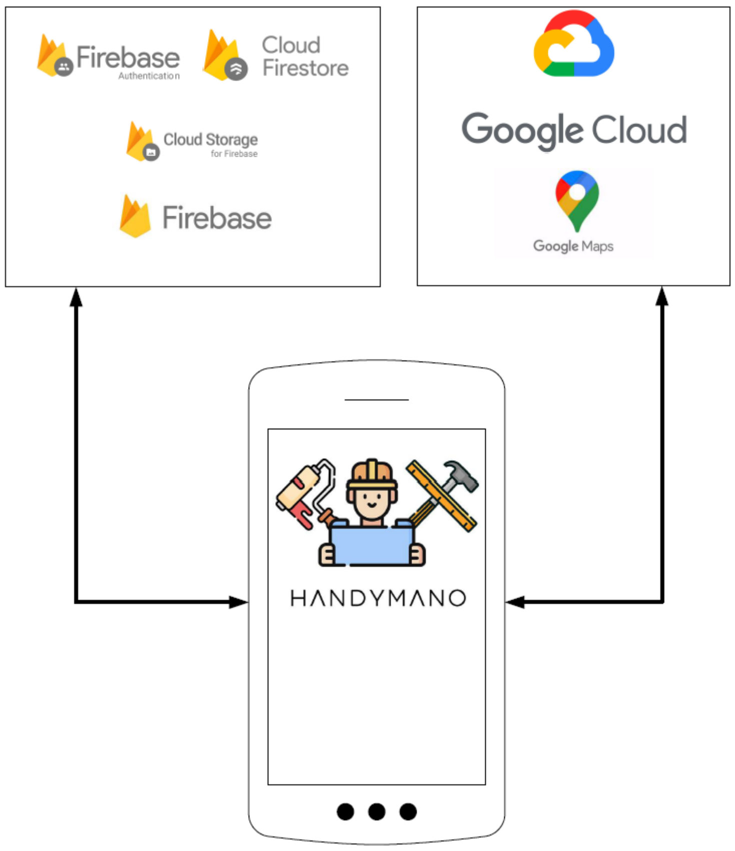


Figure 24 - overview of the infrastructure

2.2.1. Root package application activities

MainActivity

This activity is declared as the main activity in application's **Manifest.xml** and it is the first activity which is assessed when the application is started.

If the user is logged into the application then they can log into the application from this activity or if they wish to register, they can opt for the account registration.If the user is already logged into the application (checked in the *onStart()* method) then user, based on their account type, is transferred to their designated “Home” activity.

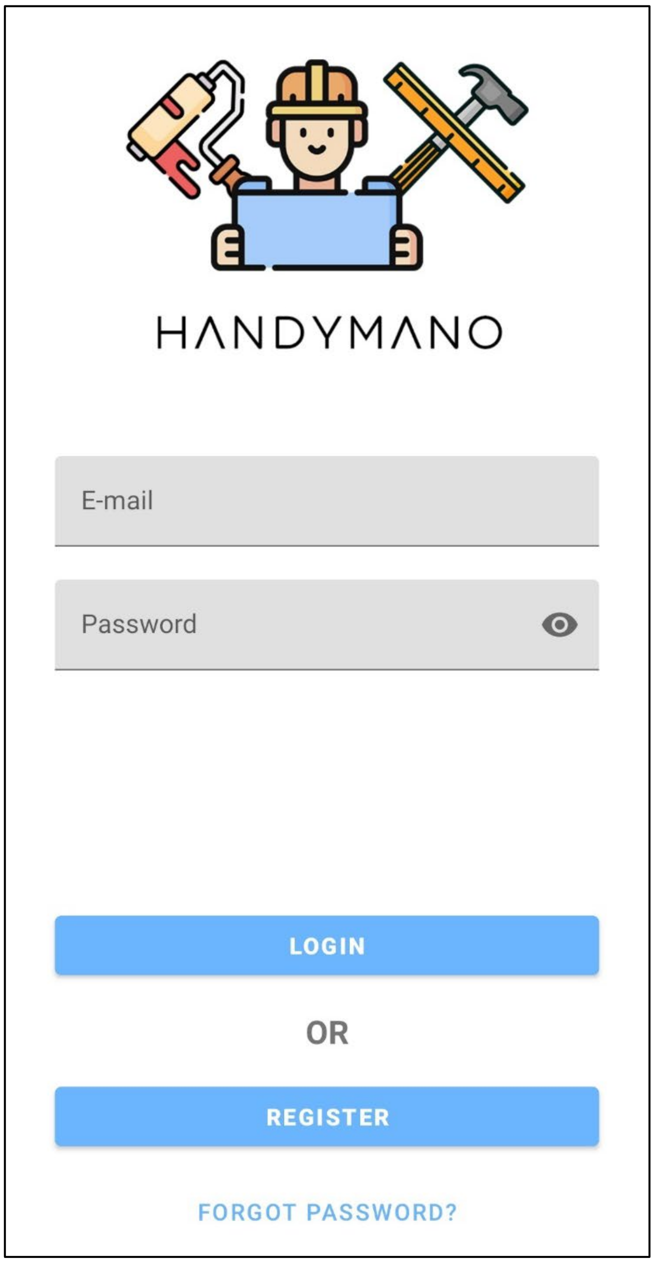


Figure 25 - login screen

RegistrationChoiceActivity

When user opts for the account creation they are transferred to this activity and they are offered to choose the type of the account creation. If the user wants to register tradesmen account, then they need to click “**I am offering services**” and they will be transferred to the “*BusinessRegistrationActivity*”. Alternatively, if the user wants to register standard account then they need to click “**I am looking for services**” and they will be transferred to the “*UserRegistrationActivity*” and lastly user can return to the “*MainActivity*” by selecting “**Cancel**” button.

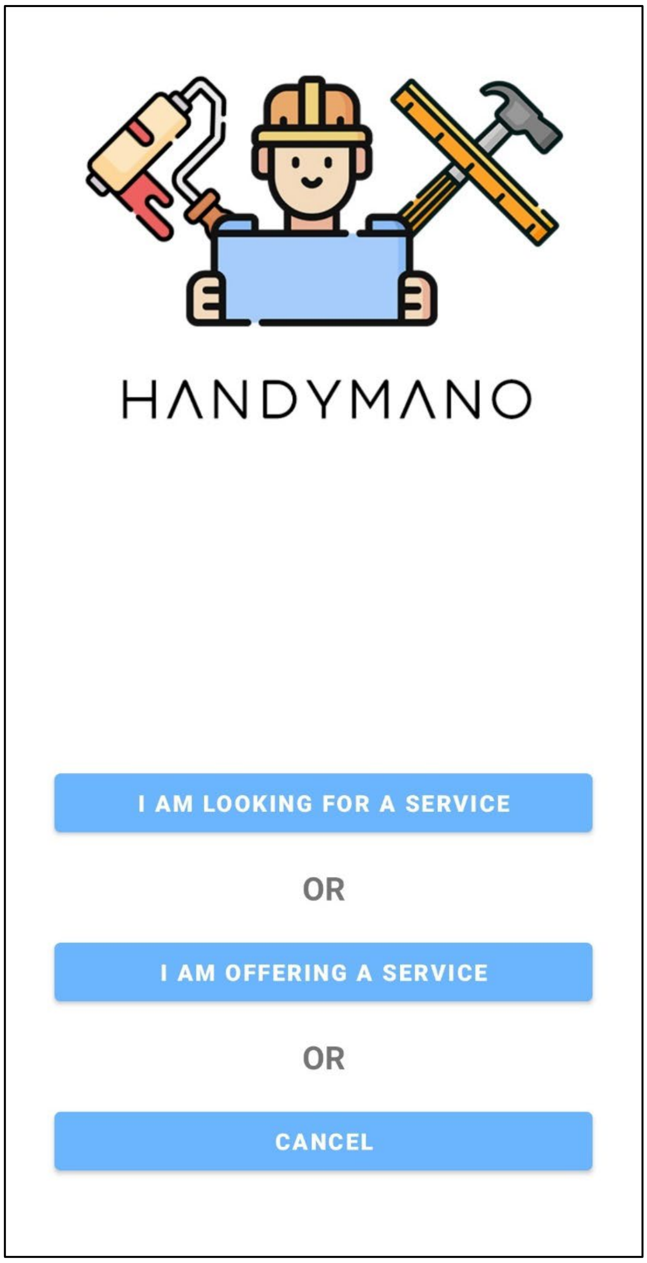


Figure 26 - registration choice

BusinessRegistrationActivity

This activity collects the following the data from the user that wants to register tradesmen account. After collecting the necessary data, if validation conditions are met, it first registers the user with Firebase authentication module and based on newly generated User ID creates new document in the Firestore "*user*" collection named after Firebase user ID with the additional user details collected during the registration.
Additional details consist of the following:

- account type

- category
- years of experience
- location data
- username
- phone number

Another two collections are created – "*rating*" and "*chat*" with a blank document named after user's Firebase user ID. Rating collection is used for storing rating and feedback data for the Professional user. Chat collection is used for storing message recipients and copies of the messages.

After this process user is transferred to the home activity.

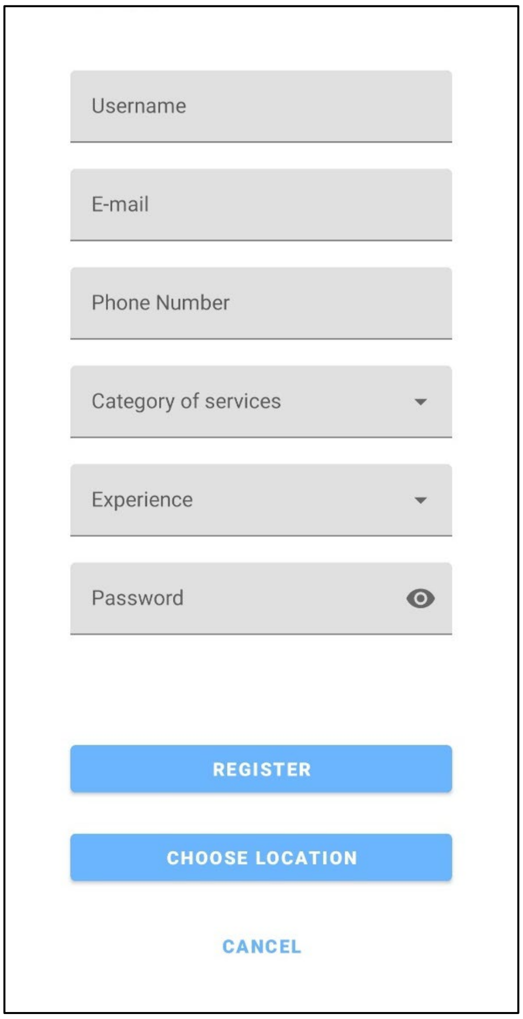
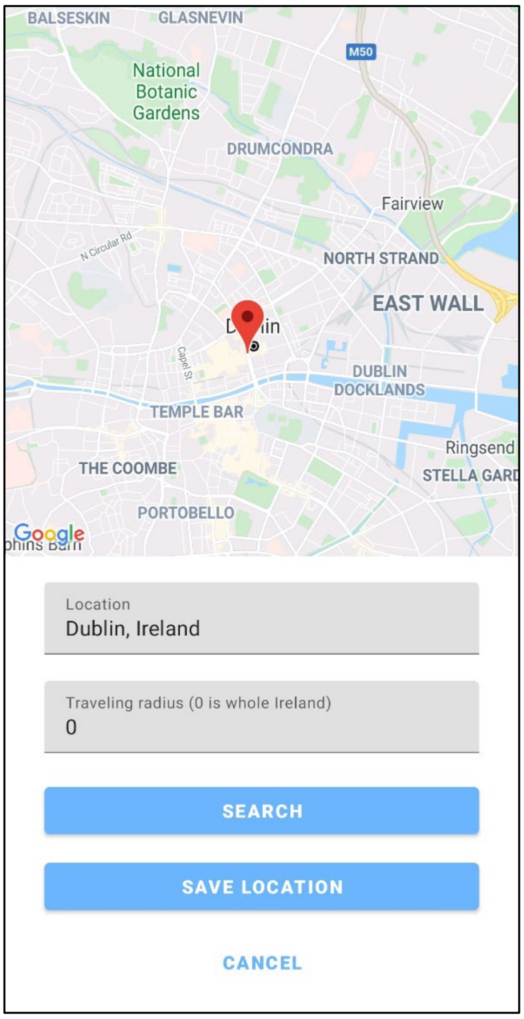


Figure 27 - professional user registration



Location
Dublin, Ireland

Traveling radius (0 is whole Ireland)
0

SEARCH

SAVE LOCATION

CANCEL

Figure 28 - professional user registration location choice

UserRegistrationActivity

Similarly to the above activity, "*UserRegistrationActivity*" collects the standard account's details, creates new account and from newly generated user ID it creates new document named after user id as well as "*chat*" collection to store message recipients array and copies of the messages.
After this process, user is transferred to the home activity.

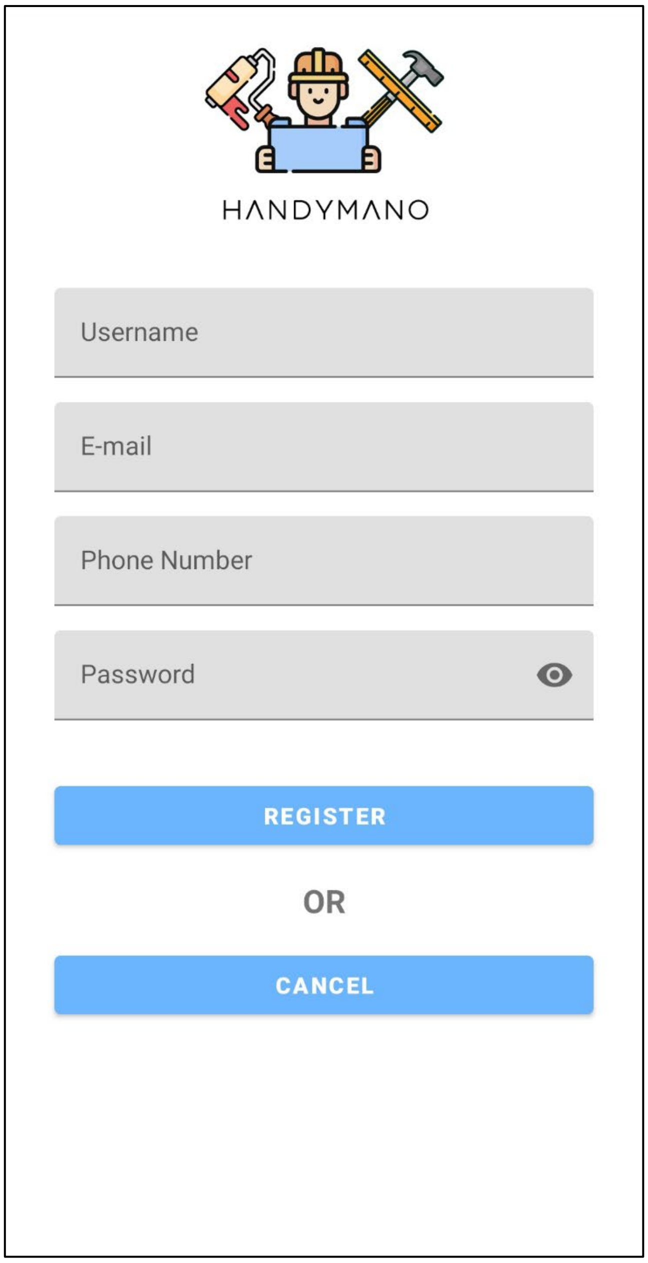


Figure 29 - standard user registration choice

PasswordReset

This activity is available from the "*MainActivity"* and user can reset their password from here.

It takes an email as a parameter and if specified email meets the criteria and exists then password reset email is sent. From there user is taken to application's Firebase project website and is instructed how to reset the password.

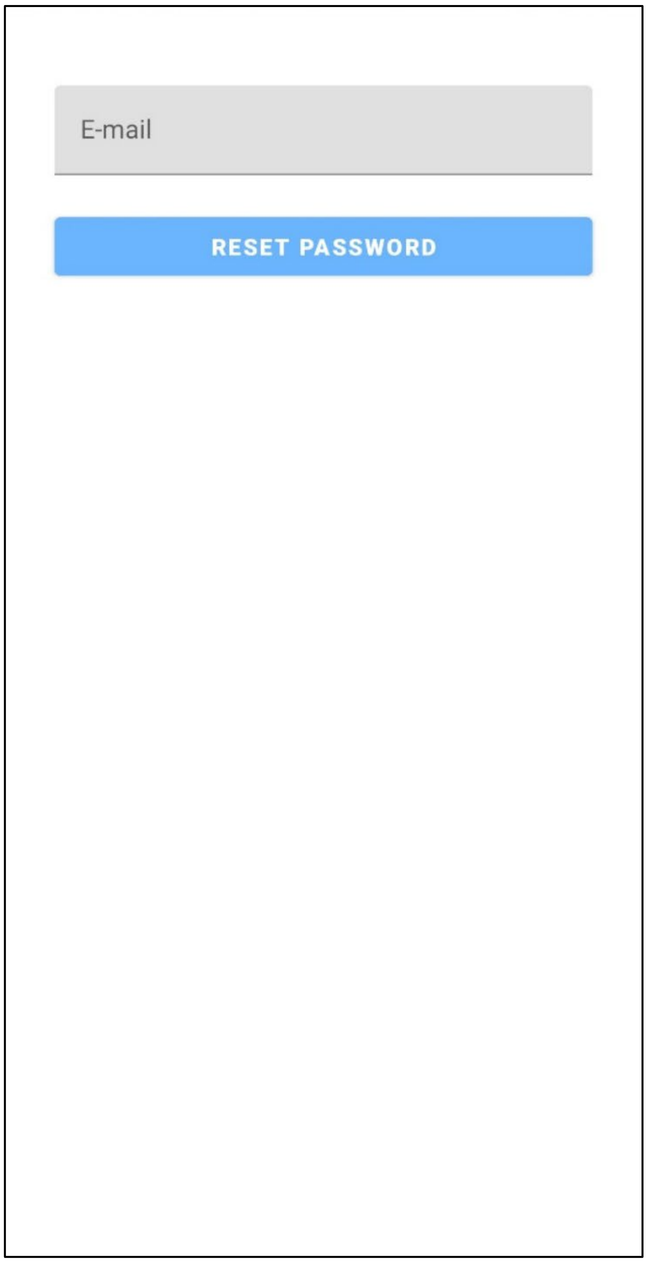


Figure 30 - password reset

MessageMenu

This activity is shared by both Standard and Professional users. When the activity is accessed it retrieves parameter “*USER_TYPE*” from the Intent to determine which Navigation Drawer layout file to use.
Activity queries current user’s document in Firestore “*chat*” collection to retrieve the list of message recipients from the “*recipients*” array.
Data in the array is structured as “***recipient_user_id,recipient_display_name***” and that array is retrieved into the local ArrayList which is then passed to the **MessagesAdapter.java** for further processing.

If the recipient array is not empty then the query iterates further through the subcollections of named after each user id from the recipient array to retrieve the last document in each subcollection based on the timestamp. These documents are queried to retrieve the last message sent in the conversation.
Then the timestamp of the last message separated by comma and followed by the last message text is added into the local ArrayList which is also passed to the **MessagesAdapter.java** for processing.
This adapter then displays the data for each recipient in the RecyclerView inside of this activity and listens for the clicks on the each RecyclerView item which leads to the **ChatActivity**.

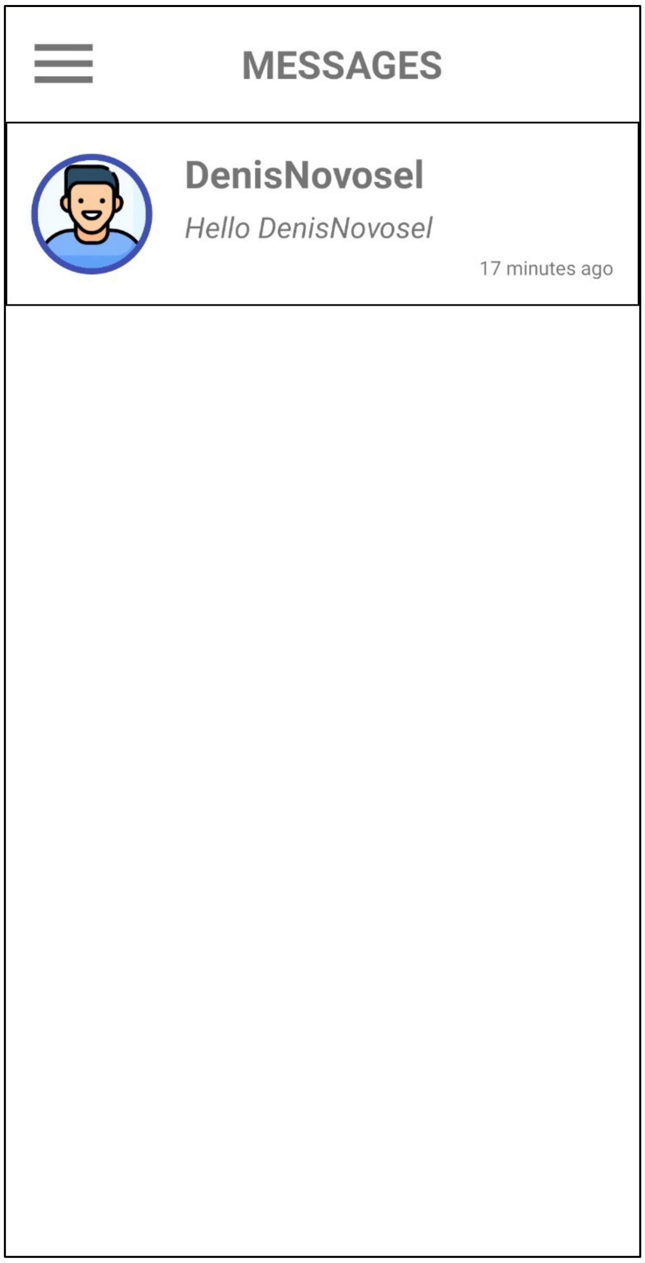


Figure 31 - message menu

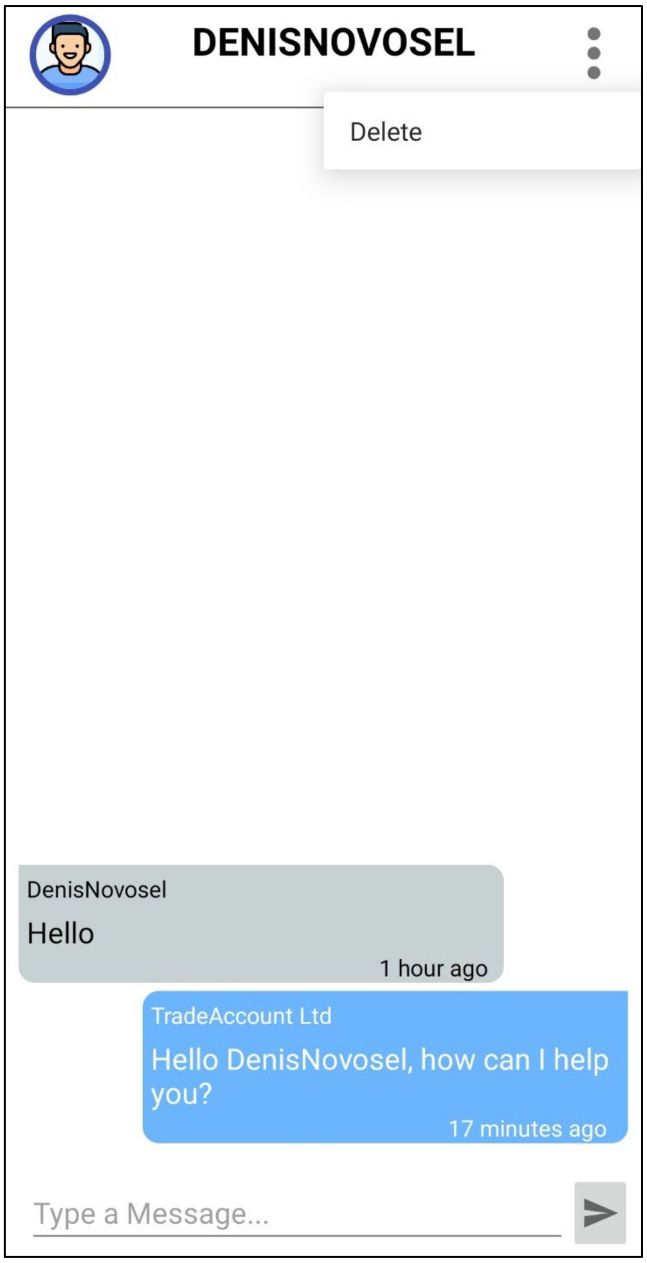
ChatActivity

Same as previous activity, it is shared by both user types. This activity can be accessed by 3 ways:

- By clicking on the *RecyclerView* item in the **MessageMenu** (both user types)
- Visiting job ad and clicking on the button “*MESSAGE ADVERTISER*” (Professional user)
- Visiting tradesman profile and clicking on the chat icon (Standard user)

Activity loads the sender and recipient’s data and checks if they exist in each other’s recipient’s array on Firestore. If they do not exist then when the message is sent new entry is created in both arrays (sender’s and receiver’s) so that the chat can be later accessed through **MessageMenu**.

When the user types the message and clicks on the send message icon application checks for the offensive content in the message. If no offensive content was found new document will be created in each of the user’s chat subcollections. Activity also offers chat deleting which will remove only one instance of the chat because we do not want to delete messages for the other user (messages are stored in both sender’s and receiver’s subcollections).



DenisNovosel
Hello
1 hour ago

TradeAccount Ltd
Hello DenisNovosel, how can I help you?
17 minutes ago

Figure 32 - chat conversation

2.2.2. Professional user application activities

HomeActivityProfessional

After user is registered and authenticated, if their account type is marked as **Professional**, then this activity acts as their dashboard.
From there they can access dashboard and navigate to different accessible parts of the application.

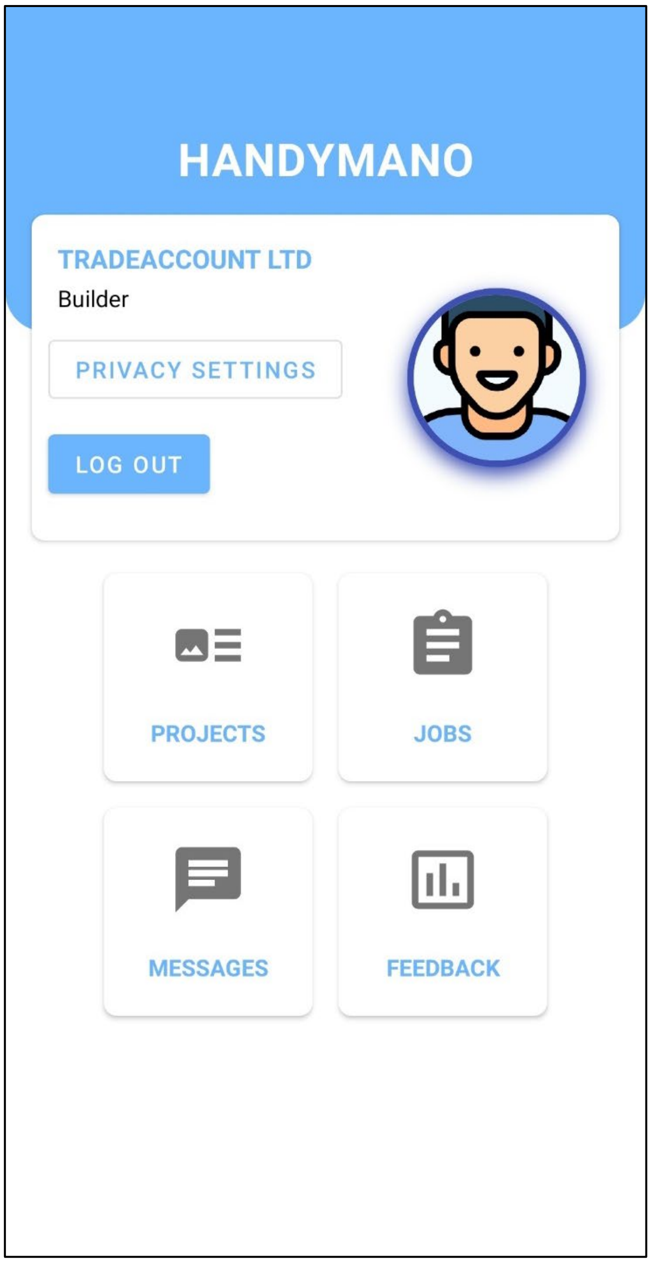
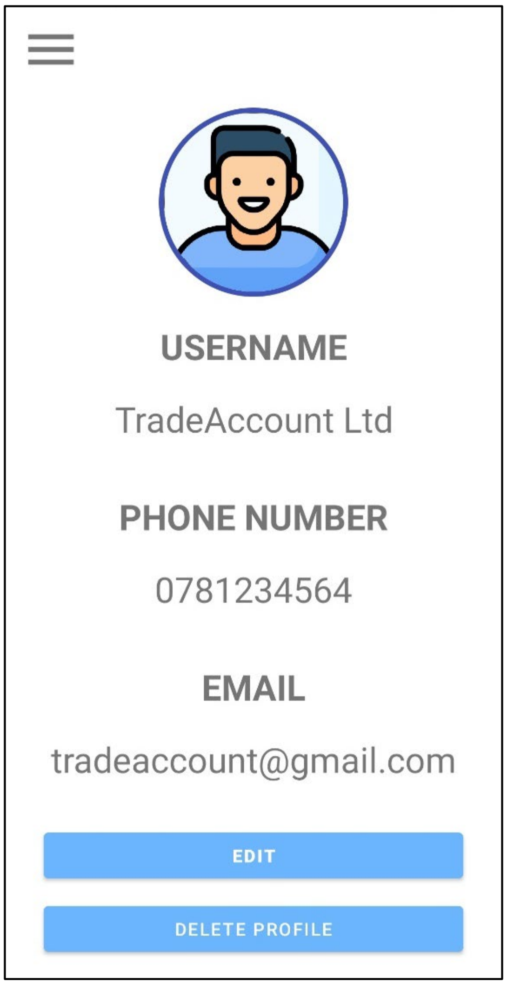


Figure 33 - professional user home screen

ProfessionalProfileActivity

This activity is available only to **Professional** users. It allows them to upload or change their profile photo as well as update their personal details. Updating personal details requires Firebase re-authentication which is handled through a password dialog.



USERNAME

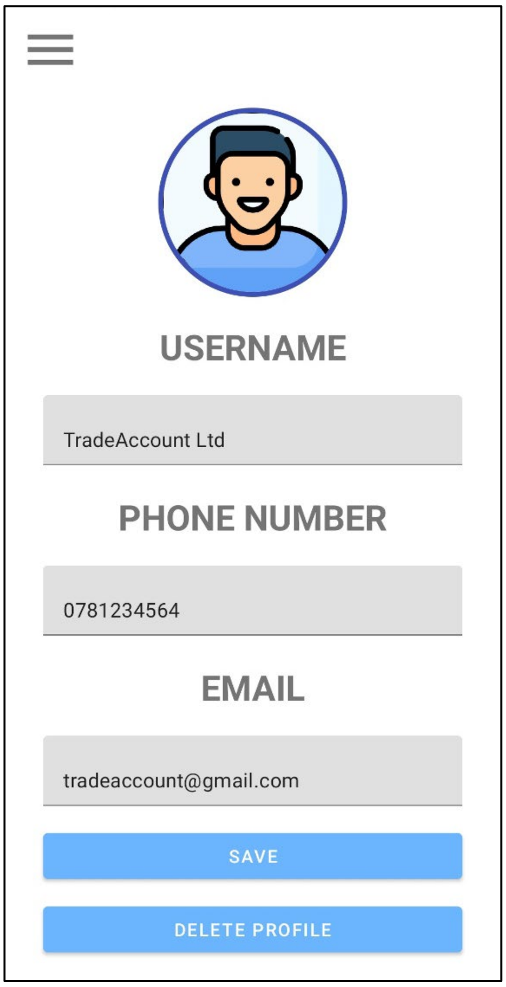
TradeAccount Ltd

PHONE NUMBER

0781234564

EMAIL

tradeaccount@gmail.com



USERNAME

TradeAccount Ltd

PHONE NUMBER

0781234564

EMAIL

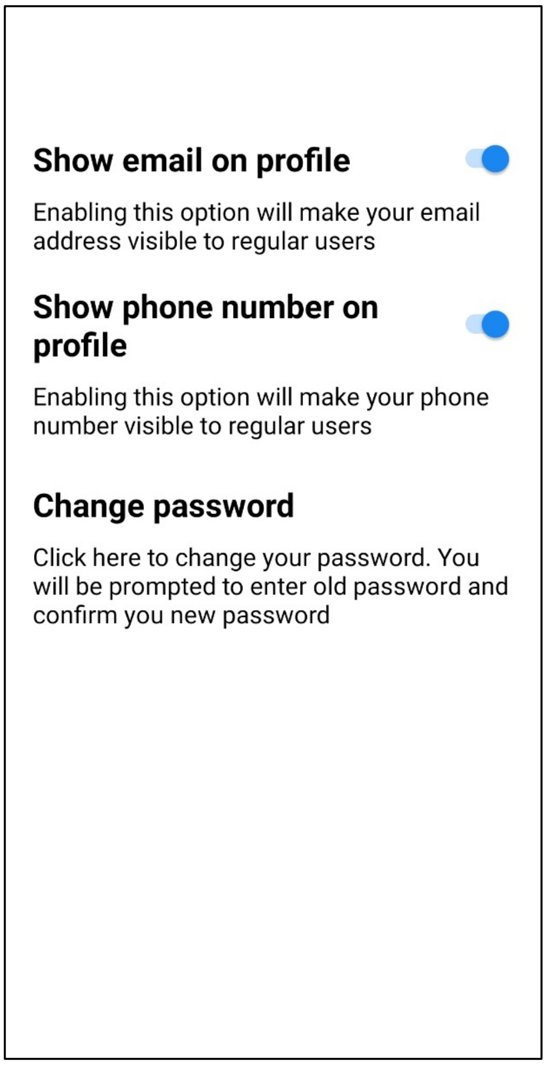
tradeaccount@gmail.com

Figure 34 - professional user profile

Figure 35 - professional user profile edit mode

PrivacySettingsActivity

Activity that is accessible only to the **Professional** users. It allows them to choose whether to make their email and phone number visible when regular user visits their profile. These options are by default set to "*false*".
Activity also allows users to change their password.



Show email on profile [✓ ON]

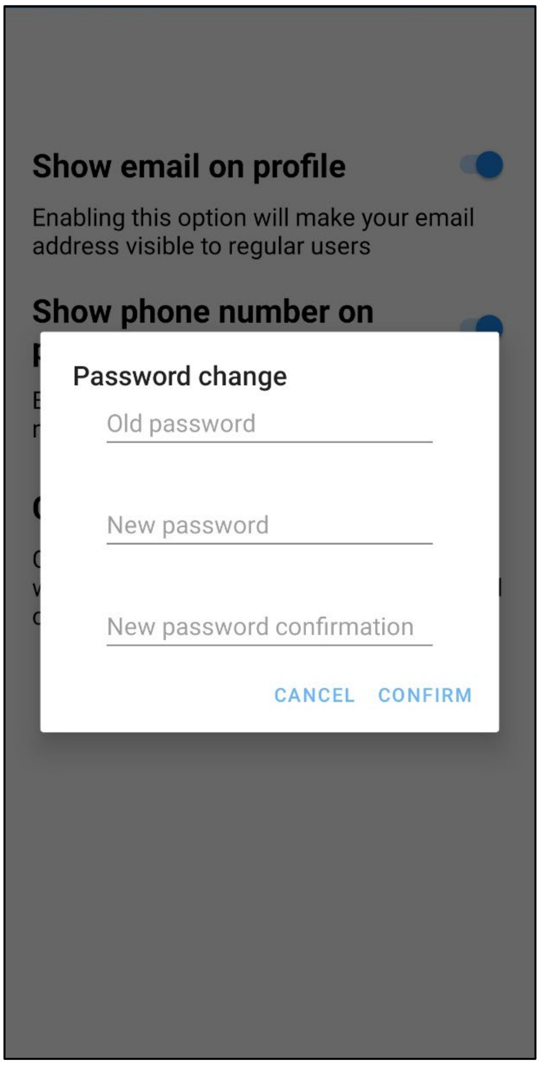
Enabling this option will make your email address visible to regular users

Show phone number on profile [✓ ON]

Enabling this option will make your phone number visible to regular users

Change password

Click here to change your password. You will be prompted to enter old password and confirm you new password



Password change

Old password

New password

New password confirmation

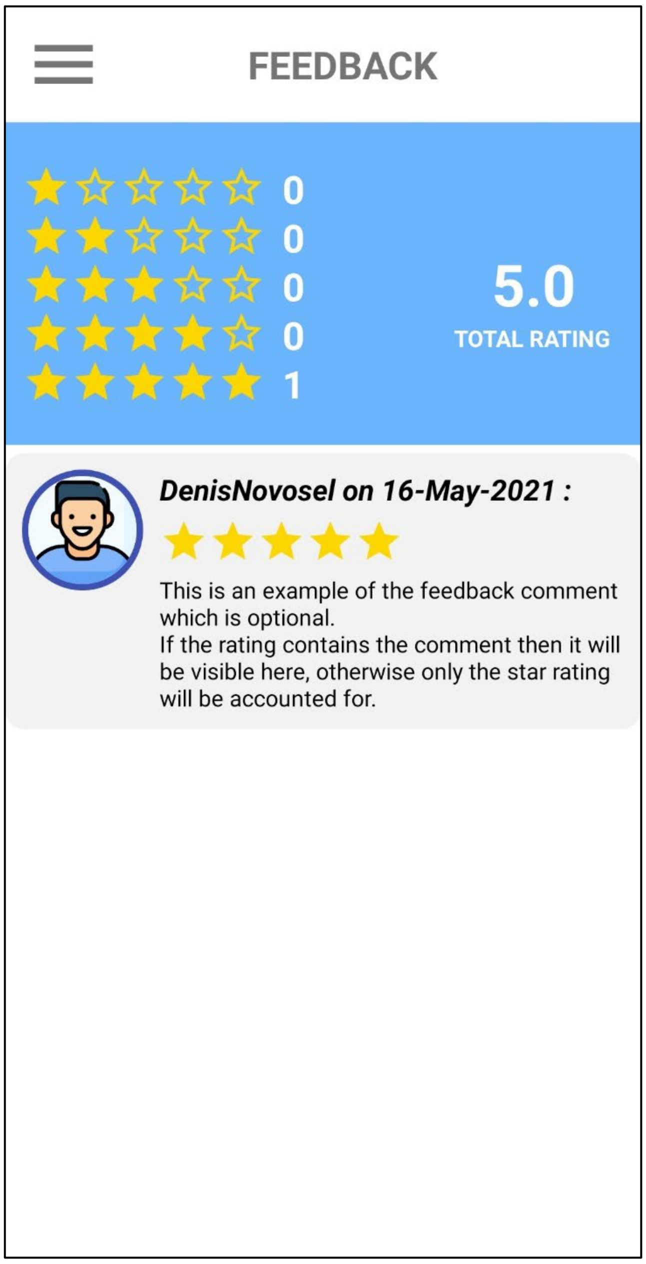
Figure 36 - professional user privacy settings

Figure 37 - professional user password change dialog

FeedbackList

This activity interacts with the Firestore collection "*rating*" and retrieves rating and feedback data based on **Professional** user's Firebase UID. After the data is retrieved from Firestore collection code within the activity calculates total rating for the specific Professional user and displays the result in the banner at the top of the activity. Ratings with feedback are shown using RecyclerView widget which shows the following:

- user who left the feedback
- individual rating
- profile image of the author
- feedback text
- date when the feedback was made



DenisNovosel on 16-May-2021 :

This is an example of the feedback comment which is optional.
If the rating contains the comment then it will be visible here, otherwise only the star rating will be accounted for.

Figure 38 - feedback list

JobsList

This activity iterates through the Firestore "*user*" collection where user type is **Standard** and checks if "*jobs*" subcollection exists. If the subcollection exists then the code checks for the following:

- if the job status is in "*Active*" mode
- if the job category matches the tradesman category
- if the job location intersects the tradesman's traveling radius

If the job matches the criteria it is then job title and the UID of the job creator is added to the ArrayList and each job name from the ArrayList is added into the RecyclerView and the each job name and UID from the ArrayList serve as a link to be used for the RecyclerView item click listener where it is passed as an extra string within intent.

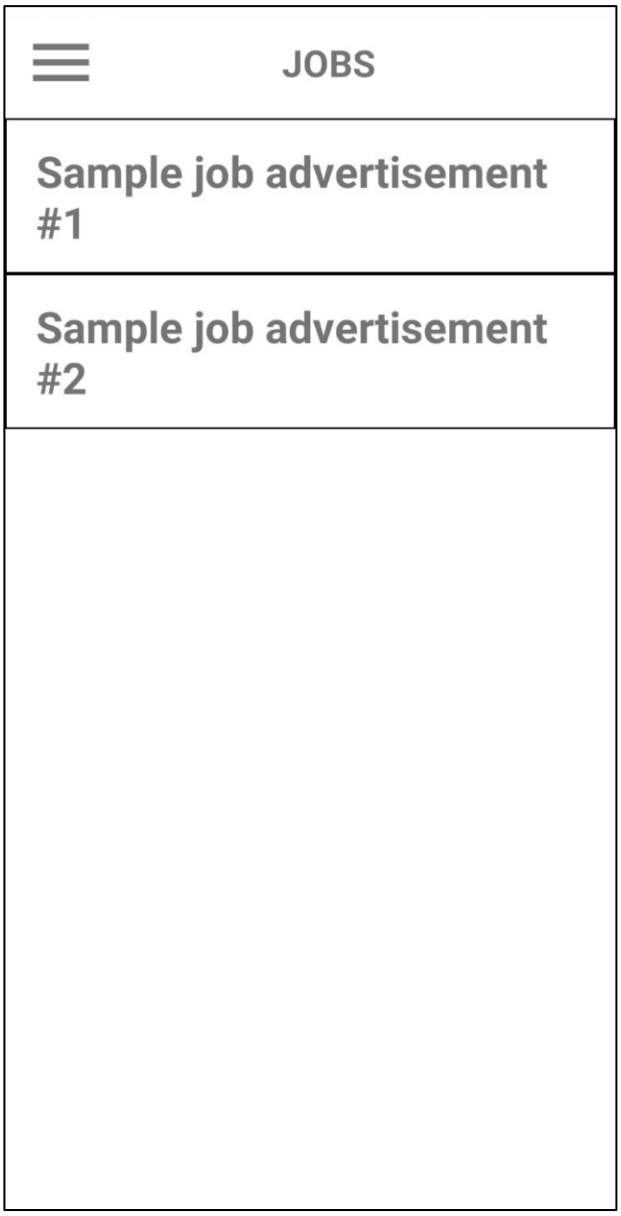
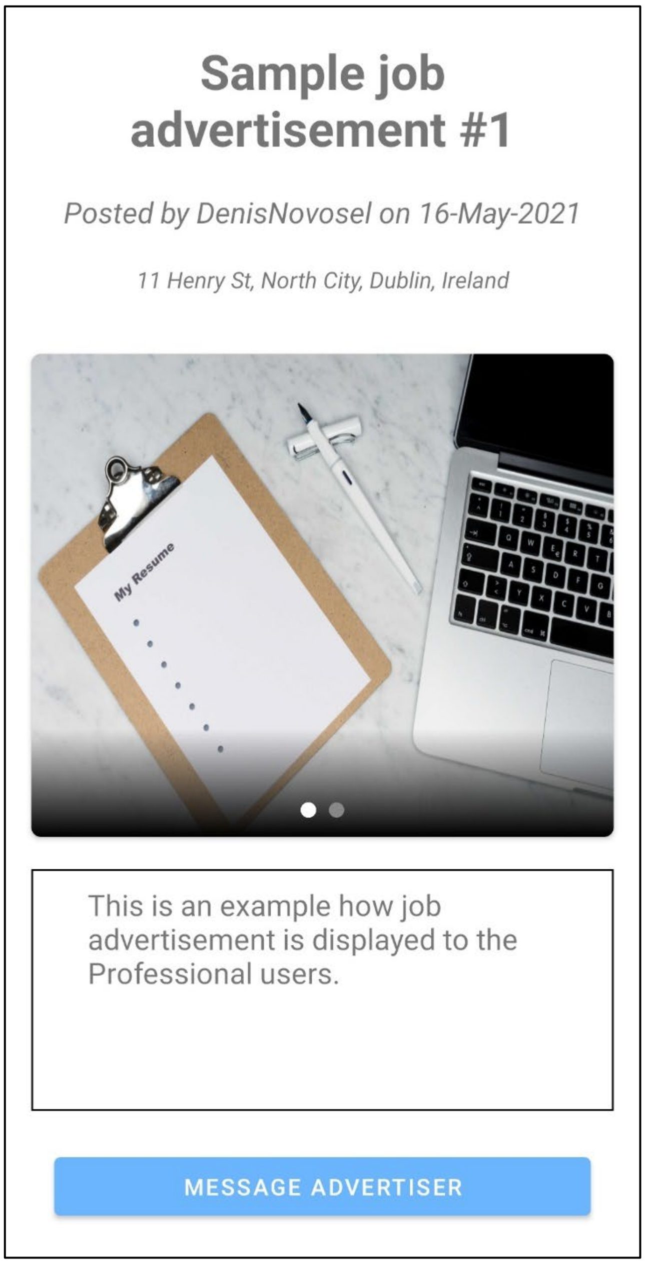


Figure 39 - job advertisement list

ViewJob

This activity retrieves job related data from Firestore document and job images from Firebase Storage based on the **UID** of the job poster and job name. Data is then shown on the activity layout and the images are loaded into image carousel. Activity displays the title of the job ad, job poster username, date when the ad was posted and the location of the job ad.



Sample job advertisement #1

Posted by DenisNovosel on 16-May-2021

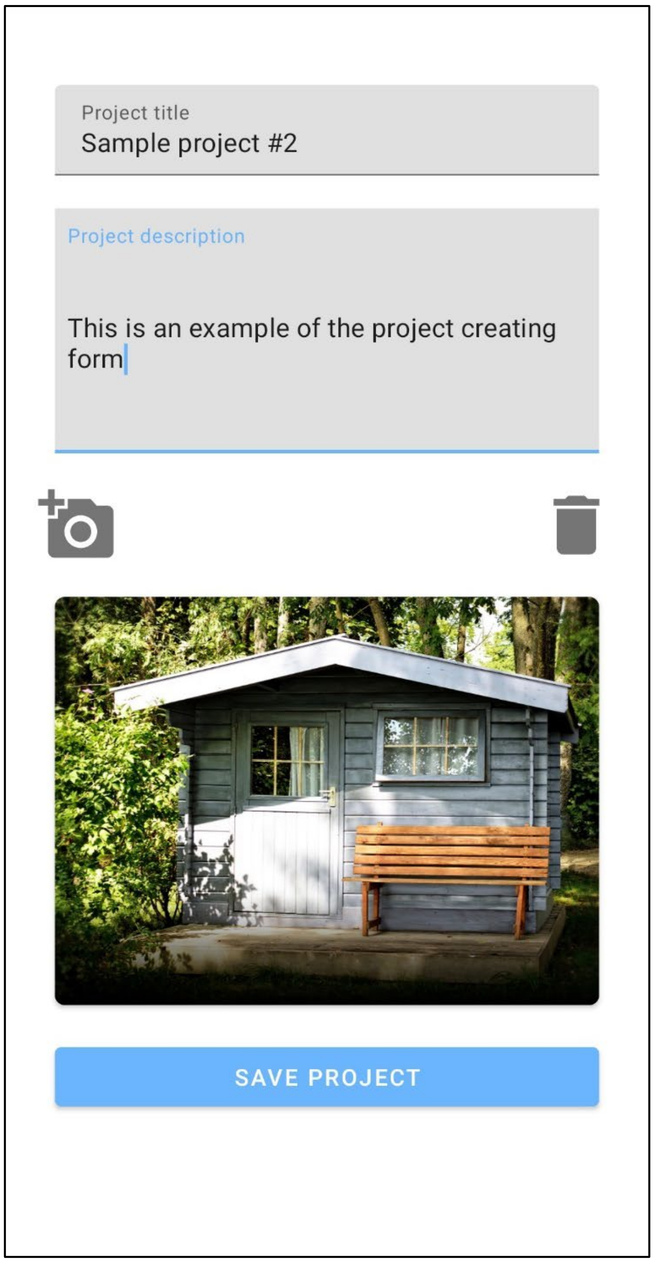
11 Henry St, North City, Dublin, Ireland

This is an example how job advertisement is displayed to the Professional users.

Figure 40 - job advertisement view

CreateProject

This activity allows tradespeople to create new project that will be available under their profile. They need to input title, description and attach at least one image to the project. When the project is saved then the project specific document is created in the "*projects*" subcollection under tradesman's profile document. Concurrently, project specific folder is created in the user's Firebase Storage folder and images are uploaded into newly created folder.

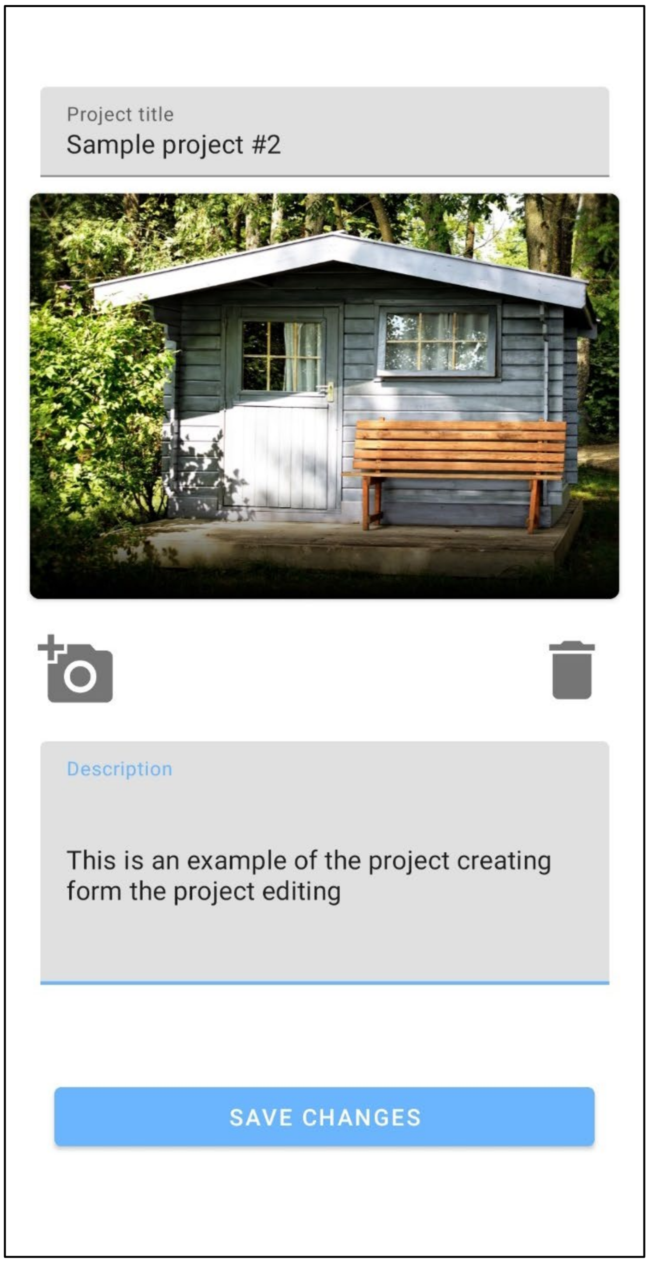

Project title
Sample project #2

Project description
This is an example of the project creating form

Figure 41 - project creation form

EditProject

This activity allows the tradespeople to edit their existing projects. Functionality is the same as in creating the project, except for the image handling. When project is being edited all the pictures assigned for that specific project are downloaded into the temporary folder on the user's phone so that they can be re-uploaded to Firebase – especially in case when the project is renamed then new directory is created on Firebase Storage and we need to upload images from somewhere.



This is an example of the project creating form the project editing

Figure 42 - project editing form

ProjectsActivity

This activity utilizes RecyclerView and FirestoreRecyclerAdapter to retrieve projects created by the Professional user that is currently logged into the application and show them in the **RecyclerView**. Each item (project) on the RecyclerView can be clicked because it utilizes "*onClickListener*" which creates new Intent that leads to the activity that displays the project.

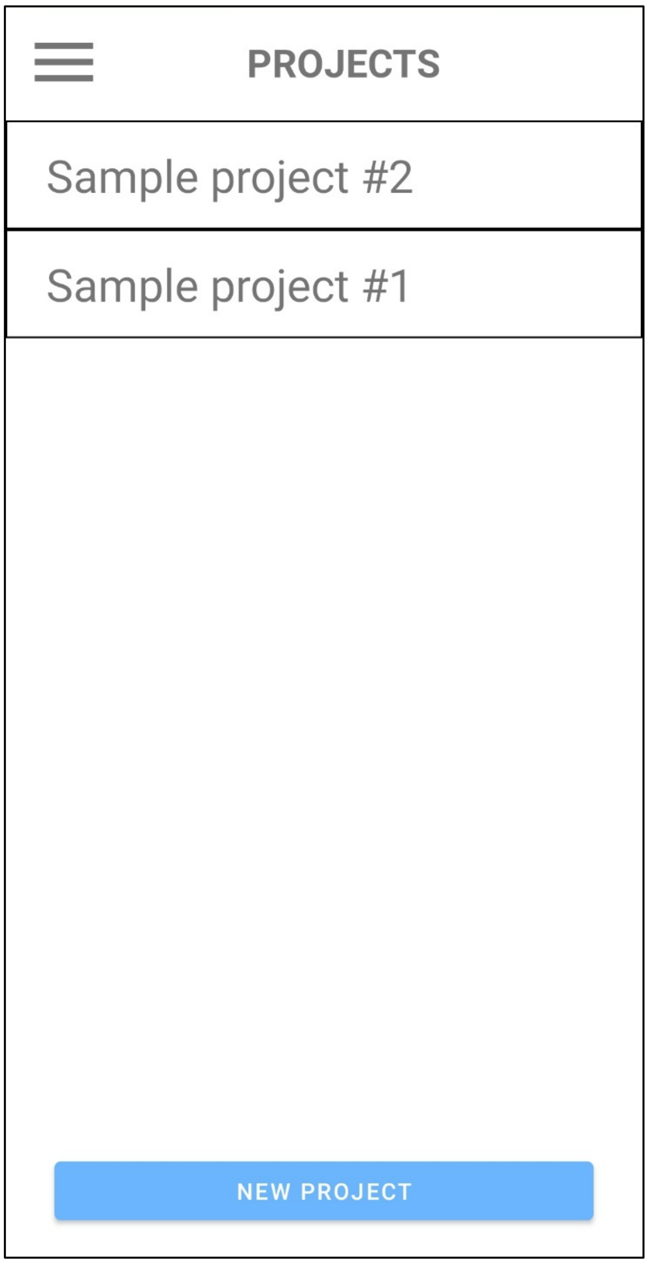
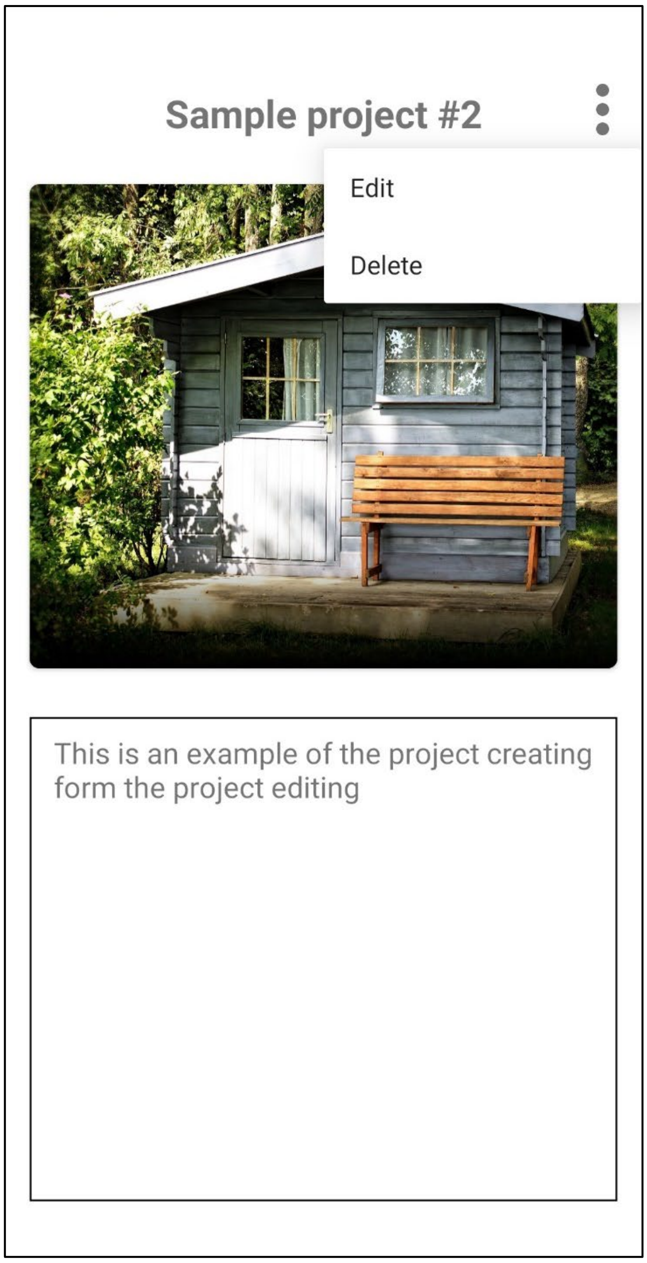


Figure 43 - project list

ViewProjectActivity

This activity is accessible only to the project creator through the "*ProjectsActivity*" described previously. Activity receives "*PROJECT_ID*" parameter from the intent and combined with the project's creator **User ID** it loads the data from Firebase Firestore and Firebase Storage. From this activity project creator can either delete or edit the project by clicking "*more*" button in the top right corner which expands the option menu.



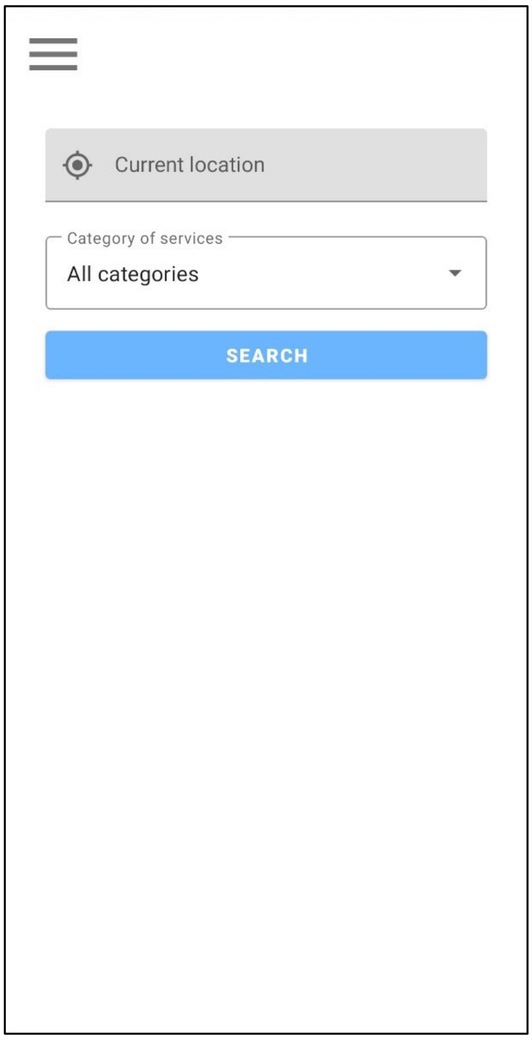
This is an example of the project creating form the project editing

Figure 44 - project view activity

2.2.3. Standard user application activities

HomeActivityStandard

This activity is available for Standard users and serves as their dashboard. From this activity user can browse for the tradesmen based on their current location or on the address specified as well as the category desired.
When the user clicks on the "*Search*" button then the activity determines location where the services are necessary. Next the method is invoked to create the temporary Firestore collection based on the results of the criteria. Criteria is that the tradesmen are in the same category that the user is browsing (or "*All categories*") and that they are willing to either travel the whole Ireland for work or that the Standard user's location is within the tradesmen's travelling radius.
After the temporary collection is populated with the results then the **FirestoreRecyclerAdapter** displays those results in the **RecyclerView**.
By clicking on the results user can visit the profile of the tradesman.



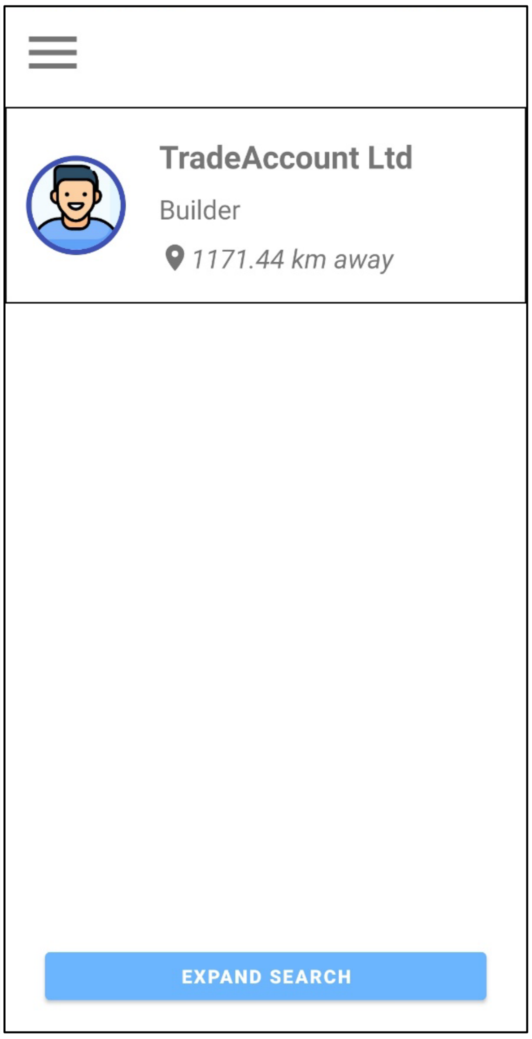


Figure 45 - standard user home activity

Figure 46 - standard user home activity result

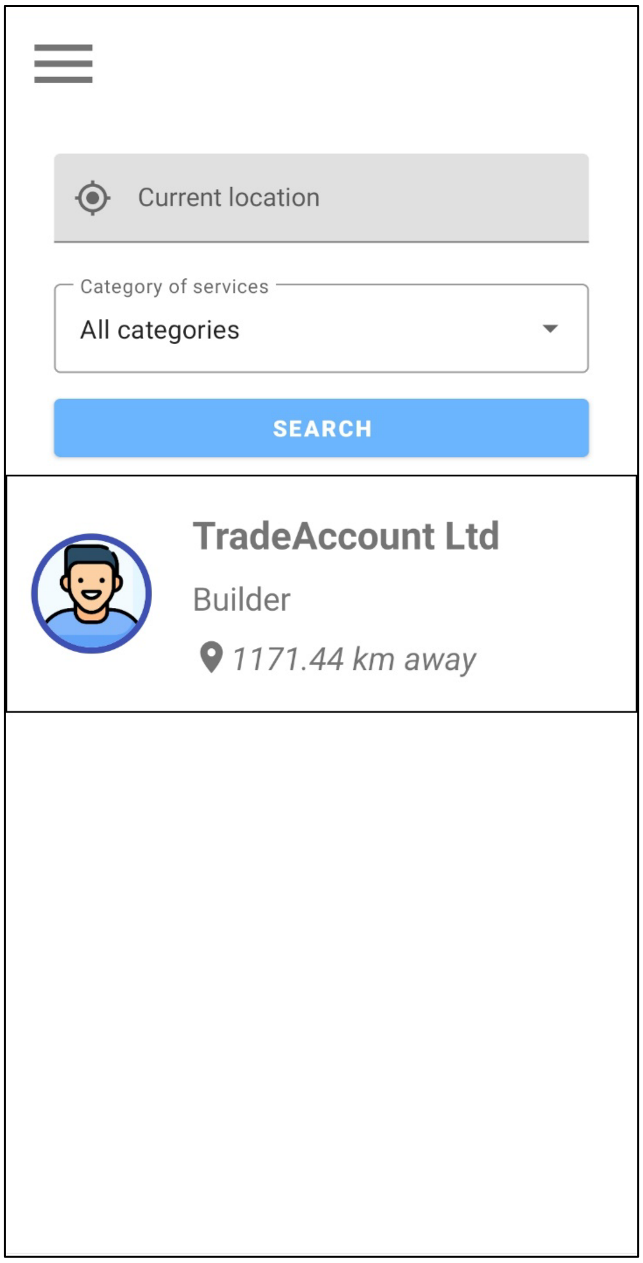
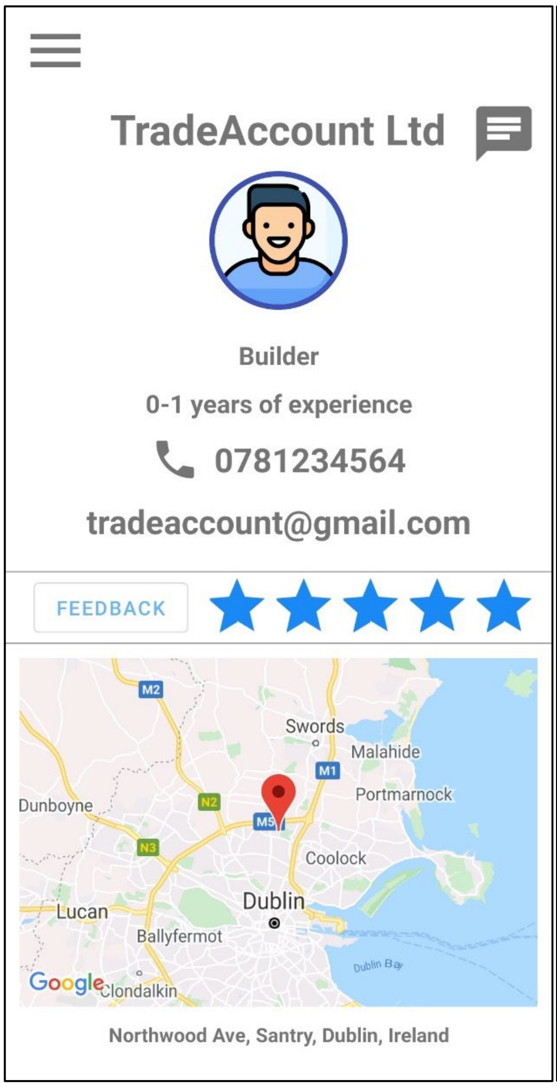


Figure 47 - home activity expanded search

ViewProfessionalActivity

From the previously explained activity Standard user can click on result inside of the RecyclerView and they will be taken to the tradesman's profile. This activity loads the data from the Firestore based on the **USER_ID** that was clicked in the previous activity. Data about the user is then display inside of the ScrollView layout.

From here the user can view their projects, their location, leave the feedback or message the user.



TradeAccount Ltd

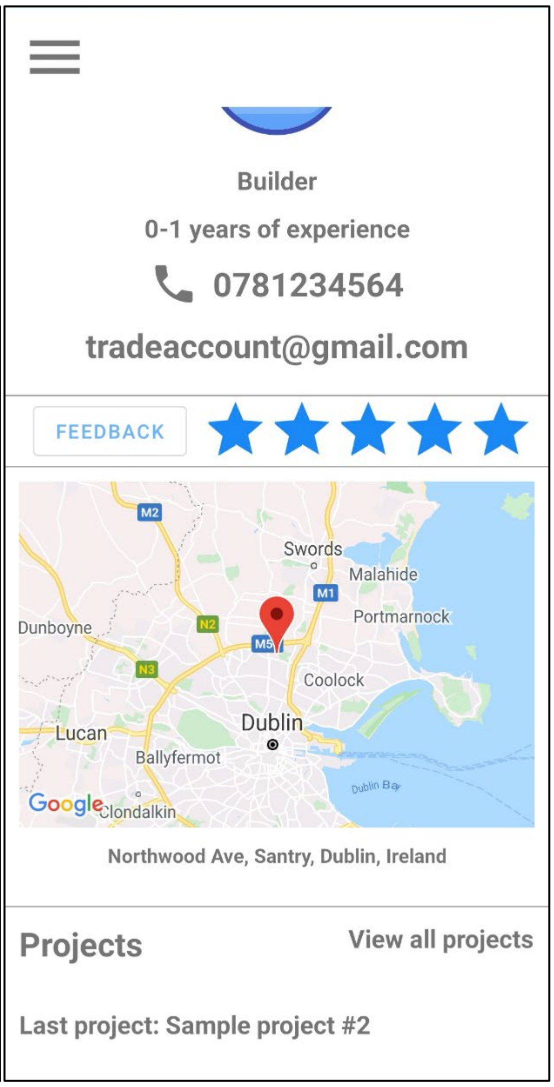
Builder

0-1 years of experience

0781234564

tradeaccount@gmail.com

Northwood Ave, Santry, Dublin, Ireland



Builder

0-1 years of experience

0781234564

tradeaccount@gmail.com

Northwood Ave, Santry, Dublin, Ireland

Projects

View all projects

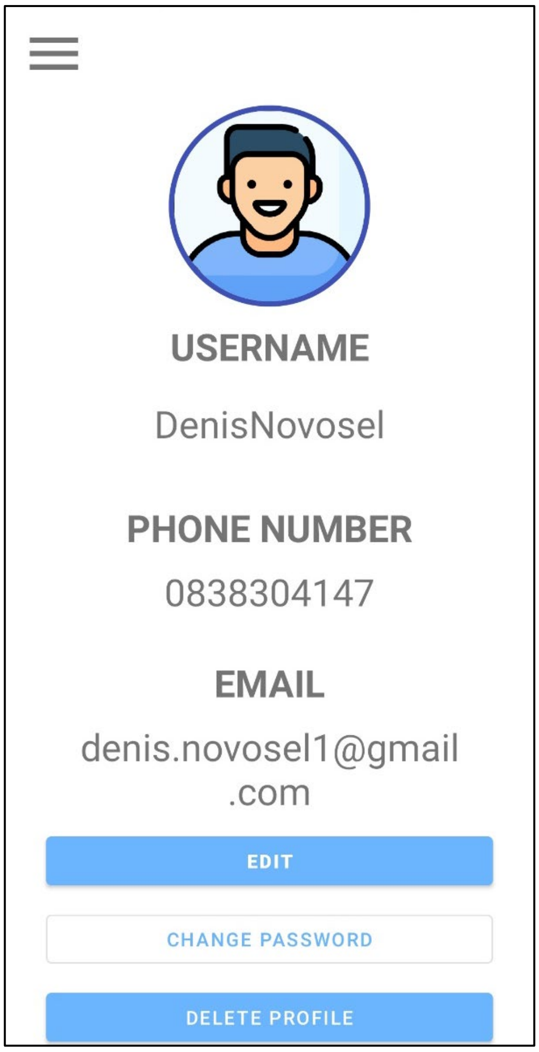
Last project: Sample project #2

Figure 48 - viewing professional user as standard user

Figure 49 - viewing professional user as standard user #2

StandardProfileActivity

This activity is available to Standard users and on it they can change their personal details. Any change requires re-authentication which is achieved by inflating custom made **PasswordConfirmationDialog** fragment.



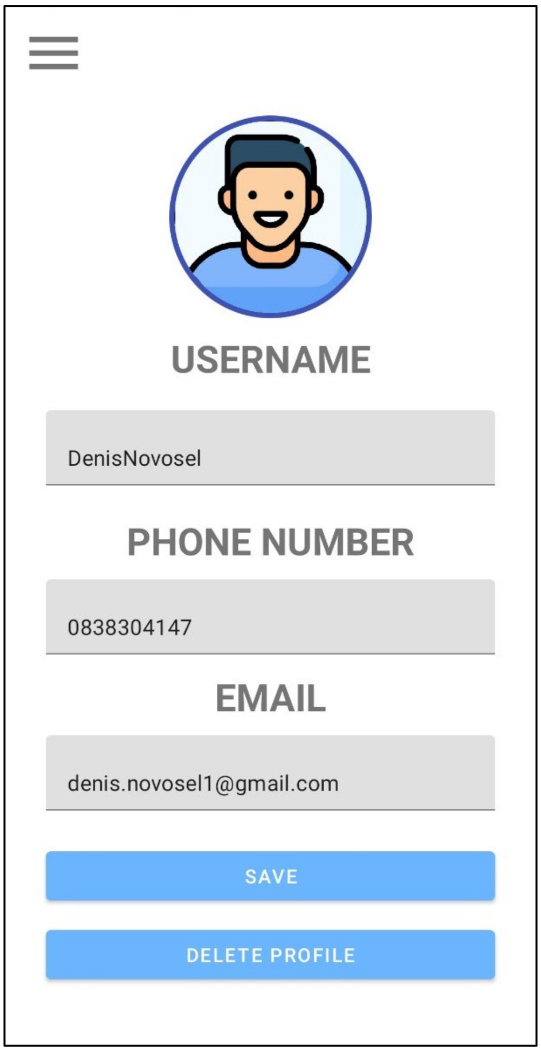
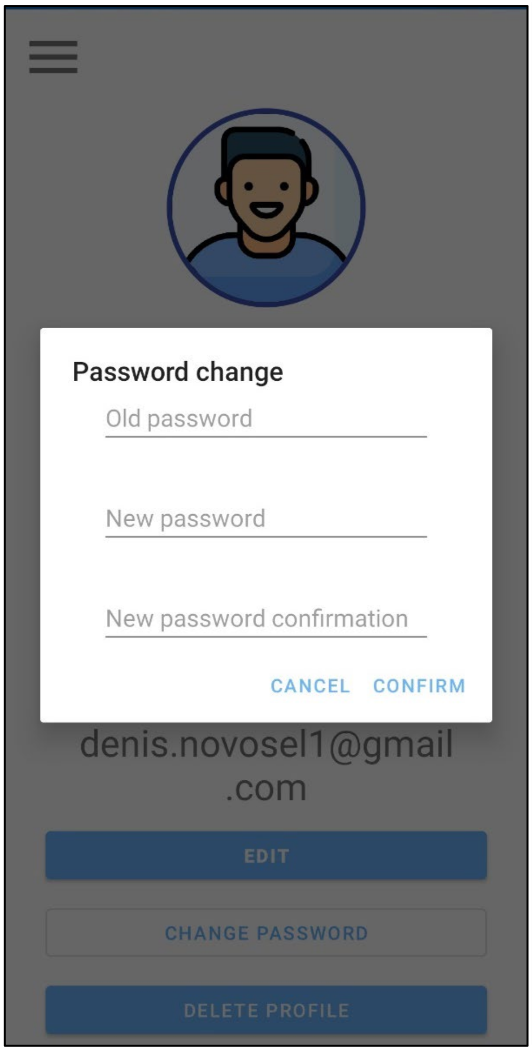


Figure 50 - standard user profile

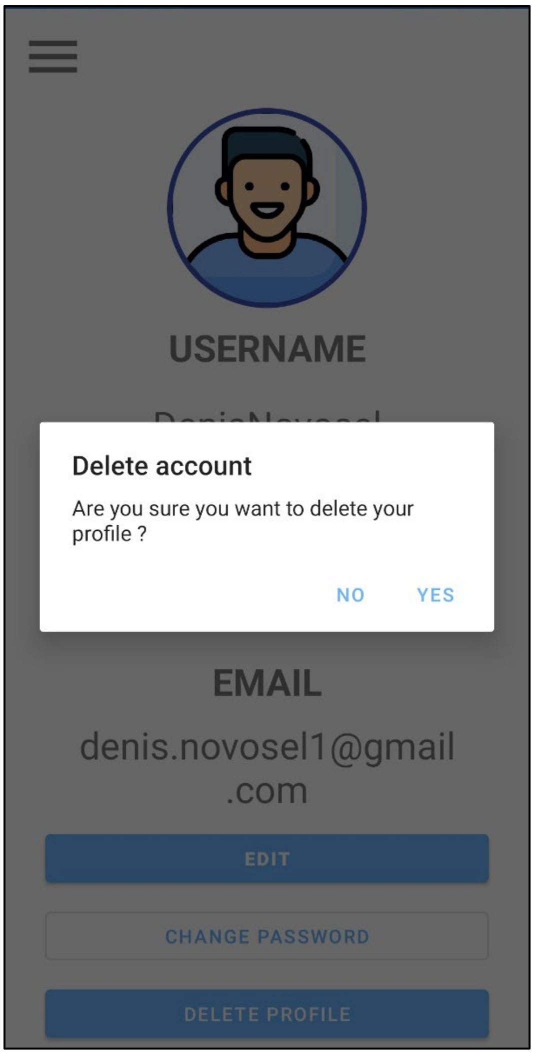
Figure 51 - standard user profile edit mode


Password change

Old password

New password

New password confirmation


Delete account

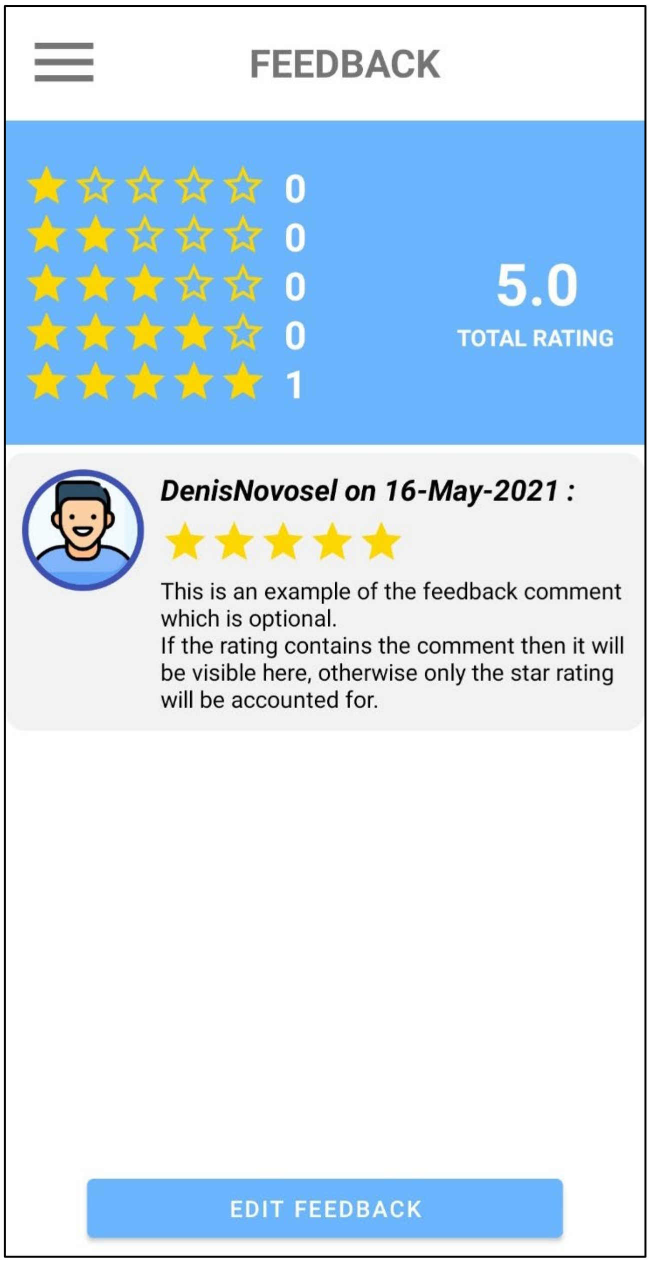
Are you sure you want to delete your profile ?

Figure 52 - password change dialog

Figure 53 - profile delete dialog

FeedbackActivity

This activity is almost identical to the FeedbackList activity which is accessible to the Professional users. Main difference is that the user has a button available which transfers them to the AddRating activity where they can add their rating and comment. If the user has not yet left the feedback about the Professional user then the button will state "***Add rating***". If the feedback or rating has been created for that user then the button will state "***Edit rating***".

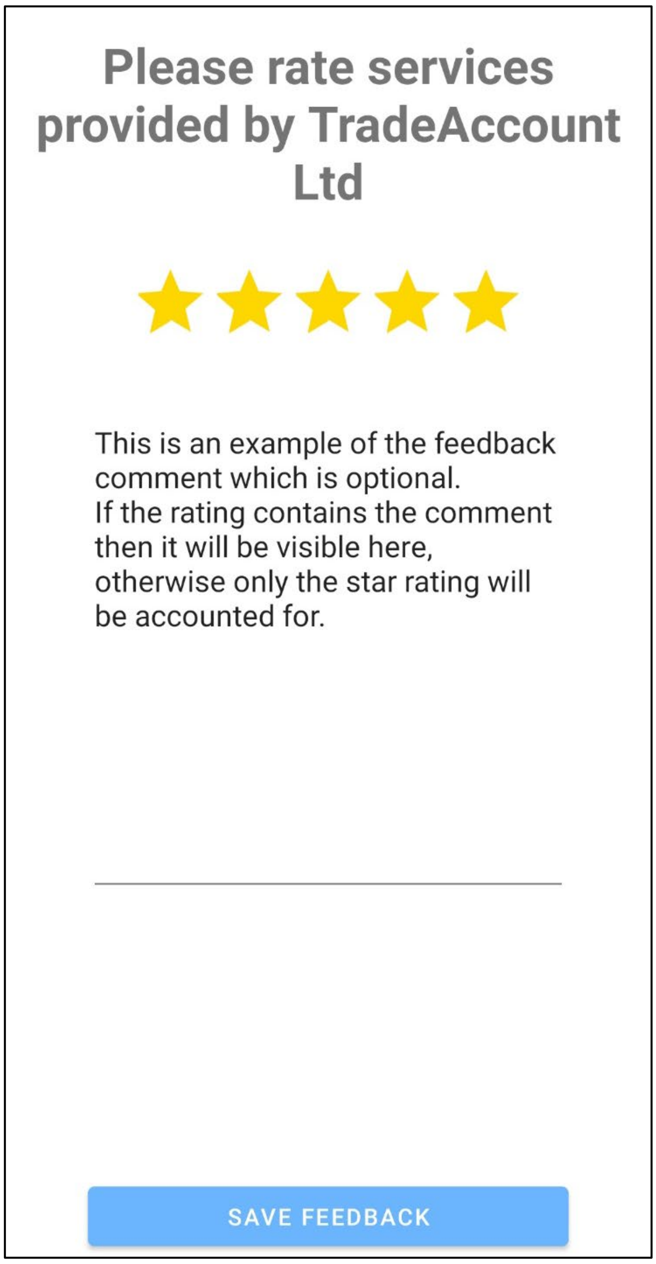


This is an example of the feedback comment which is optional.
If the rating contains the comment then it will be visible here, otherwise only the star rating will be accounted for.

Figure 54 - feedback activity

AddRating

This activity is accessible only by Standard users through FeedbackActivity. In this activity the user can choose to rate the Professional user 1-5 stars and optionally leave the feedback. If the feedback comment is not provided then the rating will not appear in the list of the feedback, but the rating will still be accounted in the total rating.

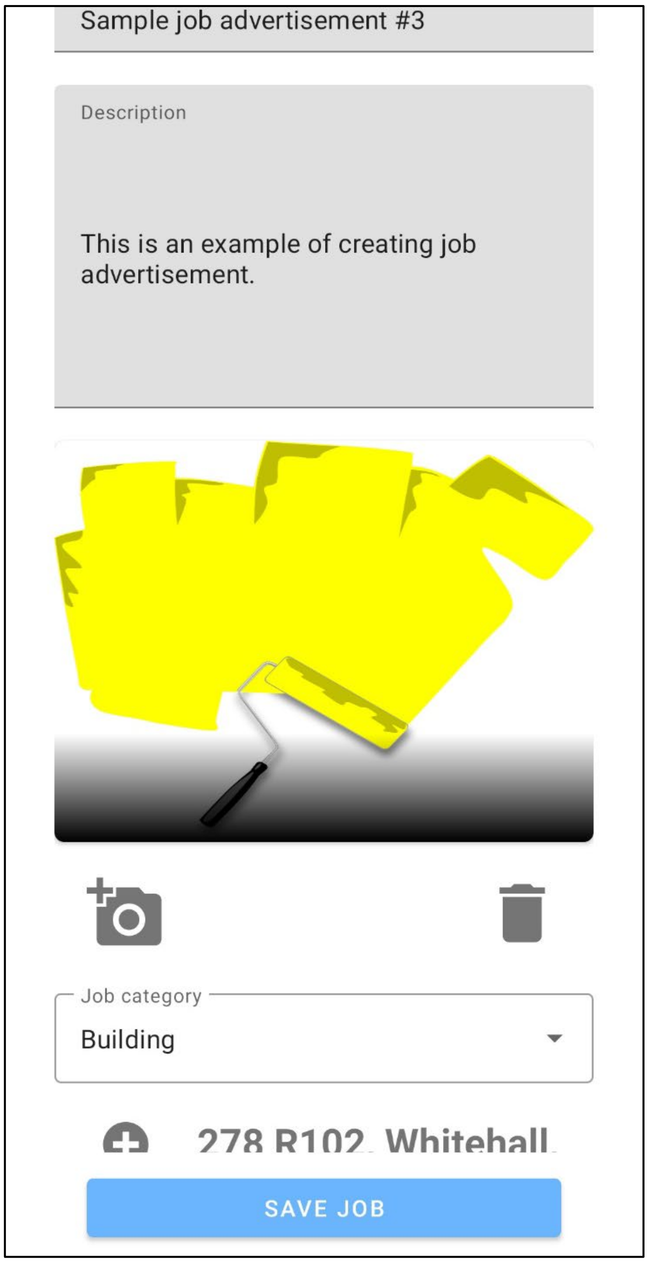

Please rate services provided by TradeAccount Ltd

This is an example of the feedback comment which is optional.
If the rating contains the comment then it will be visible here, otherwise only the star rating will be accounted for.

Figure 55 - feedback creating activity

CreateJob

On this activity Standard users can create job ads which contain title, description, category, location and images. These ads will be visible to tradesmen which are in the same category and whose traveling radius engulfs the job location.



Sample job advertisement #3

Description

This is an example of creating job advertisement.

Job category

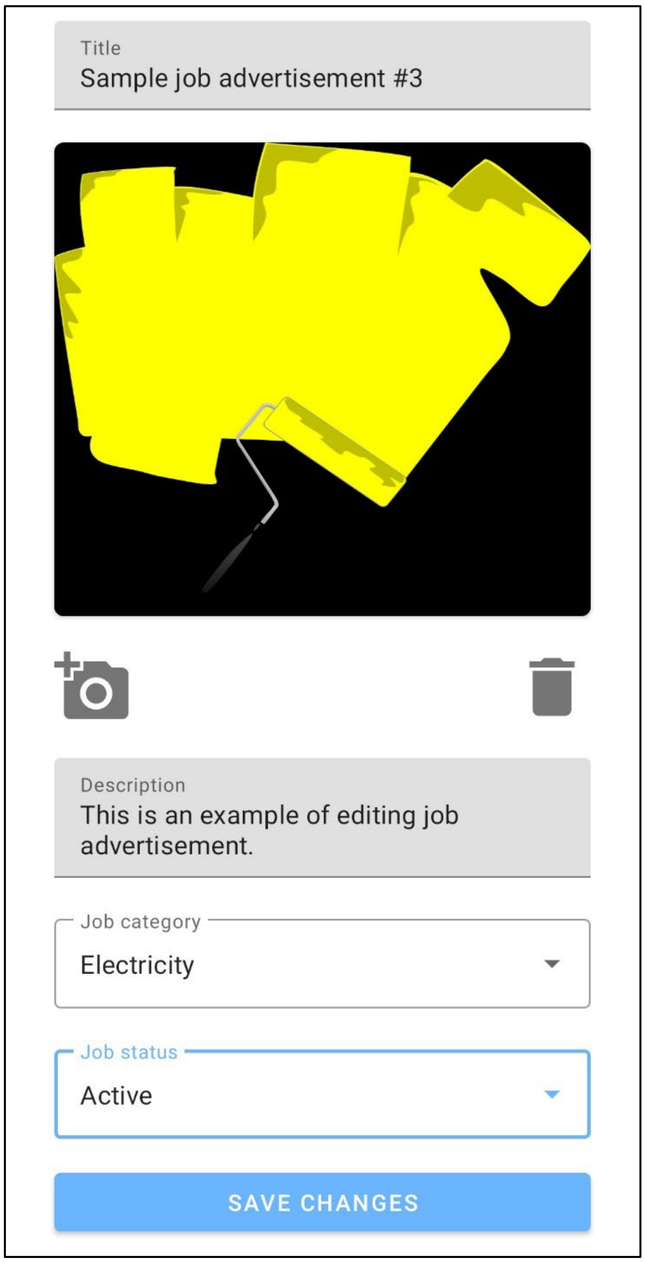
Building

278 R102, Whitehall

Figure 56 - job advertisement create mode

EditJob

This activity is accessed through the **StandardJobViewActivity** and it retrieves the **JOB_ID** parameter from the Intent to retrieve job ad data from the Firestore.
User can edit all the data on the ad and change the status (Active or Private). If the status is set to Private then it will only be visible to the job ad creator.



Title
Sample job advertisement #3

Description
This is an example of editing job advertisement.

Job category
Electricity

Job status
Active

Figure 57 - job advertisement edit mode

JobsActivity

This activity displays all the jobs created by the user in both Private and Active statuses within the **RecyclerView**. Click on each item will transfer the user to the StandardJobViewActivity and it will pass the job ID as a parameter that is used to retrieve job ad data.

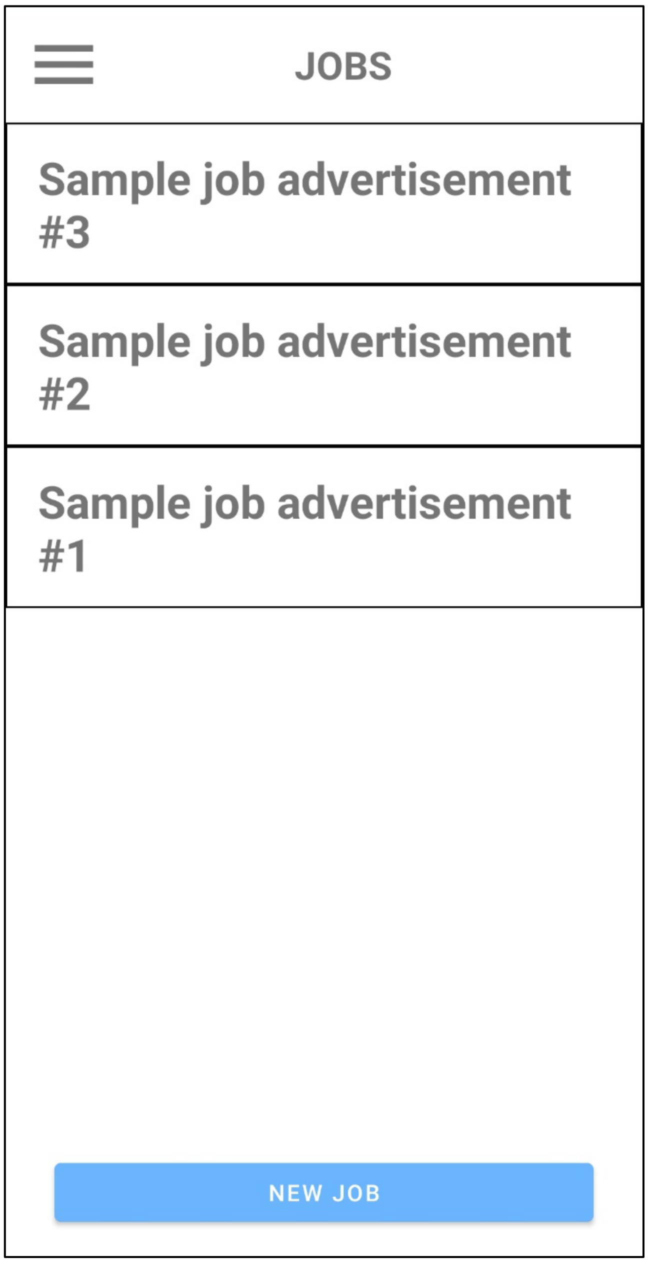
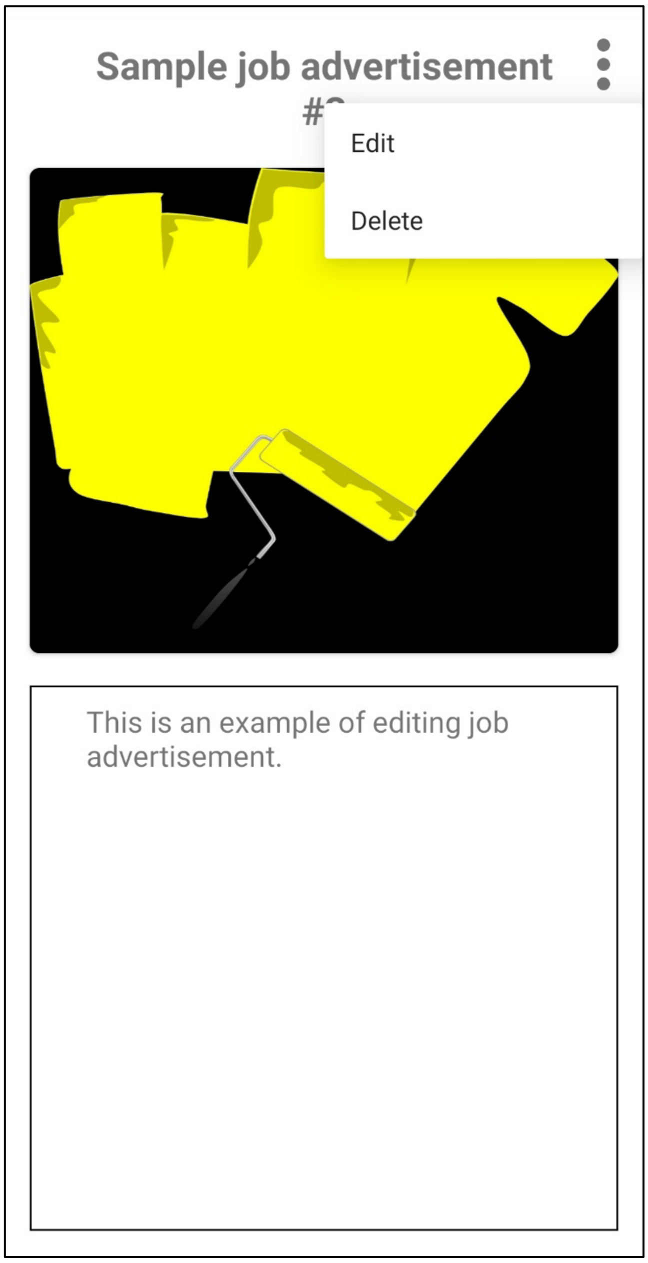

JOBS
Sample job advertisement #3
Sample job advertisement #2
Sample job advertisement #1
NEW JOB

Figure 58 - job advertisement list

StandardJobViewActivity

When this activity is access then the **JOB_ID** parameter is received from the intent and it is used to query Firestore to display the job ad data within the activity's layout. Activity also contains button to expand the options which are Edit and Delete. If the user clicks on "***Edit***" then they are taken to the **EditJob** activity where they can change the job ad details and if they click on "***Delete***" then they will be asked to confirm deletion of the job ad. If positive button is selected then the job will be deleted and user is transferred to the JobsActivity.



This is an example of editing job advertisement.

Figure 59 - job advertisement view

ProjectList

When the Standard user runs **ViewProfessionalActivity**, besides viewing tradesman's details they can also view all their Projects by clicking "***View all projects***". This button will be visible only if the Professional user has projects created. Clicking this button will lead the user to the ProjectList activity which loads all the Professional user's projects into the **RecyclerView**. Clicking any items in this RecyclerView leads the user to the ViewProject activity.

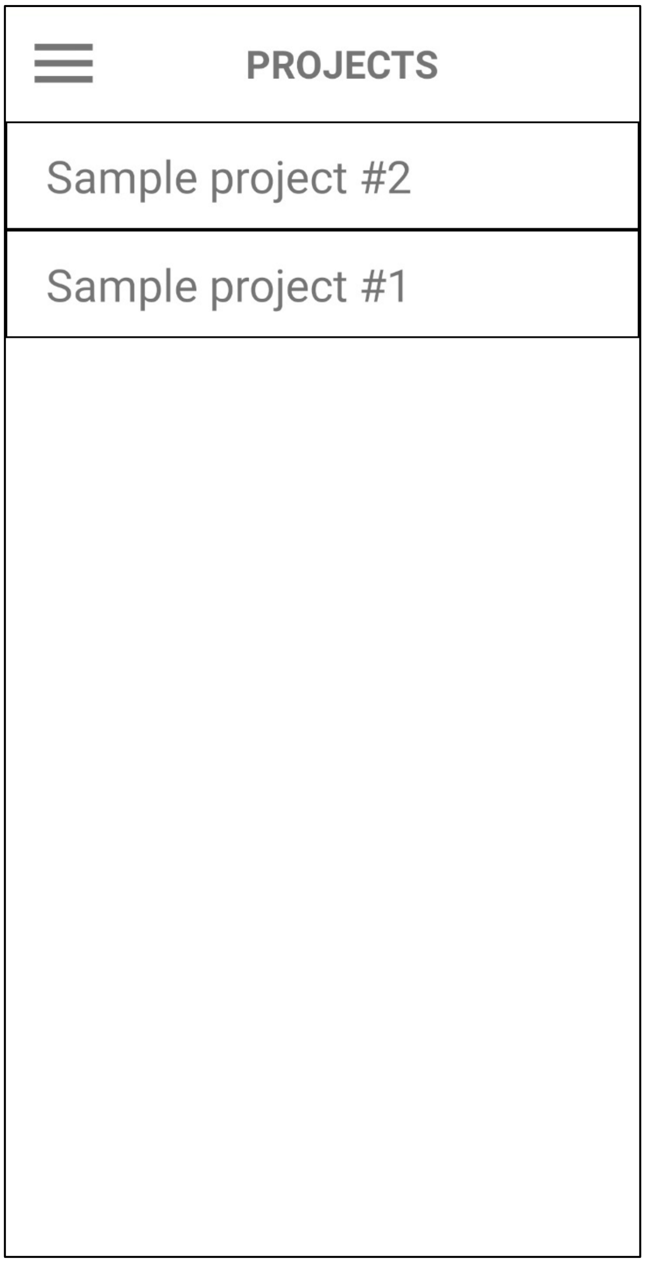
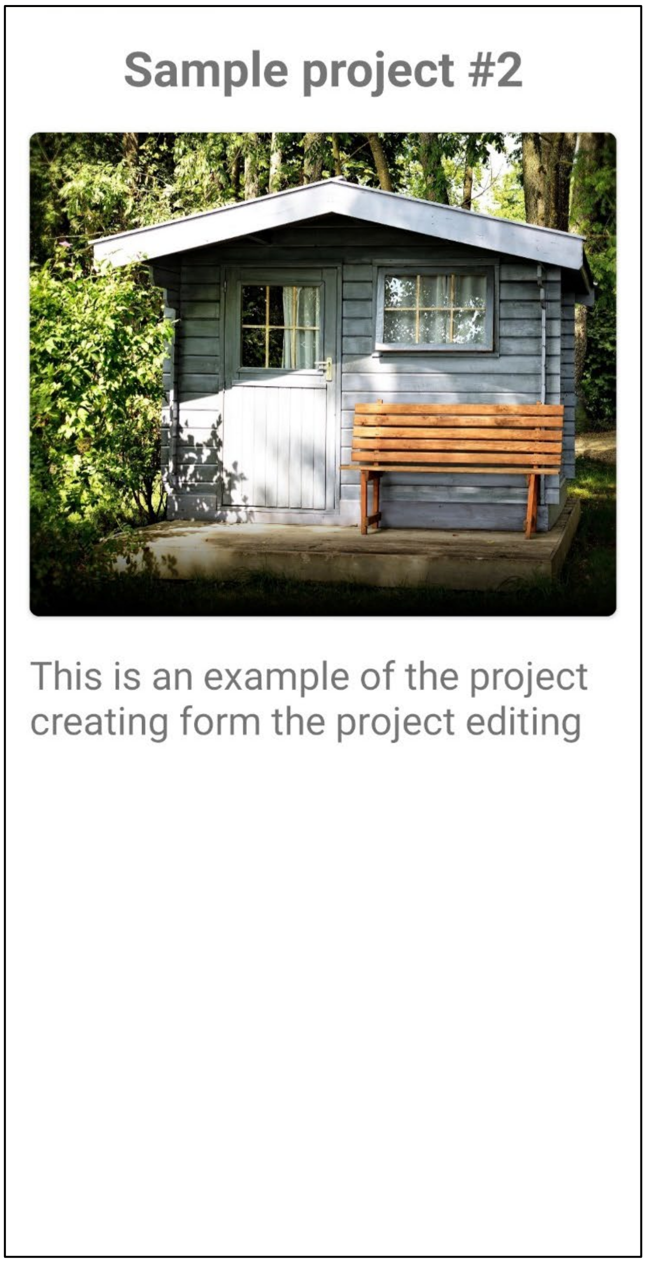


Figure 60 - project list

ViewProject

Standard user can visit this activity by either selecting one of the projects from the ProjectList or clicking on the title of the last project created by the Professional user from their profile in ViewProfessionalActivity. In both cases **USER_ID** and **PROJECT_ID** parameters are passed and retrieved from the intent on this activity so that the activity can query Firebase Firestore and Storage. Data retrieved is displayed in the activity's layout.



Sample project #2

This is an example of the project creating form the project editing

Figure 61 - project view

2.3. Implementation

2.3.1. Naming convention

Throughout my code base I have tried to hold onto the consistent naming of variables, objects and layout items.

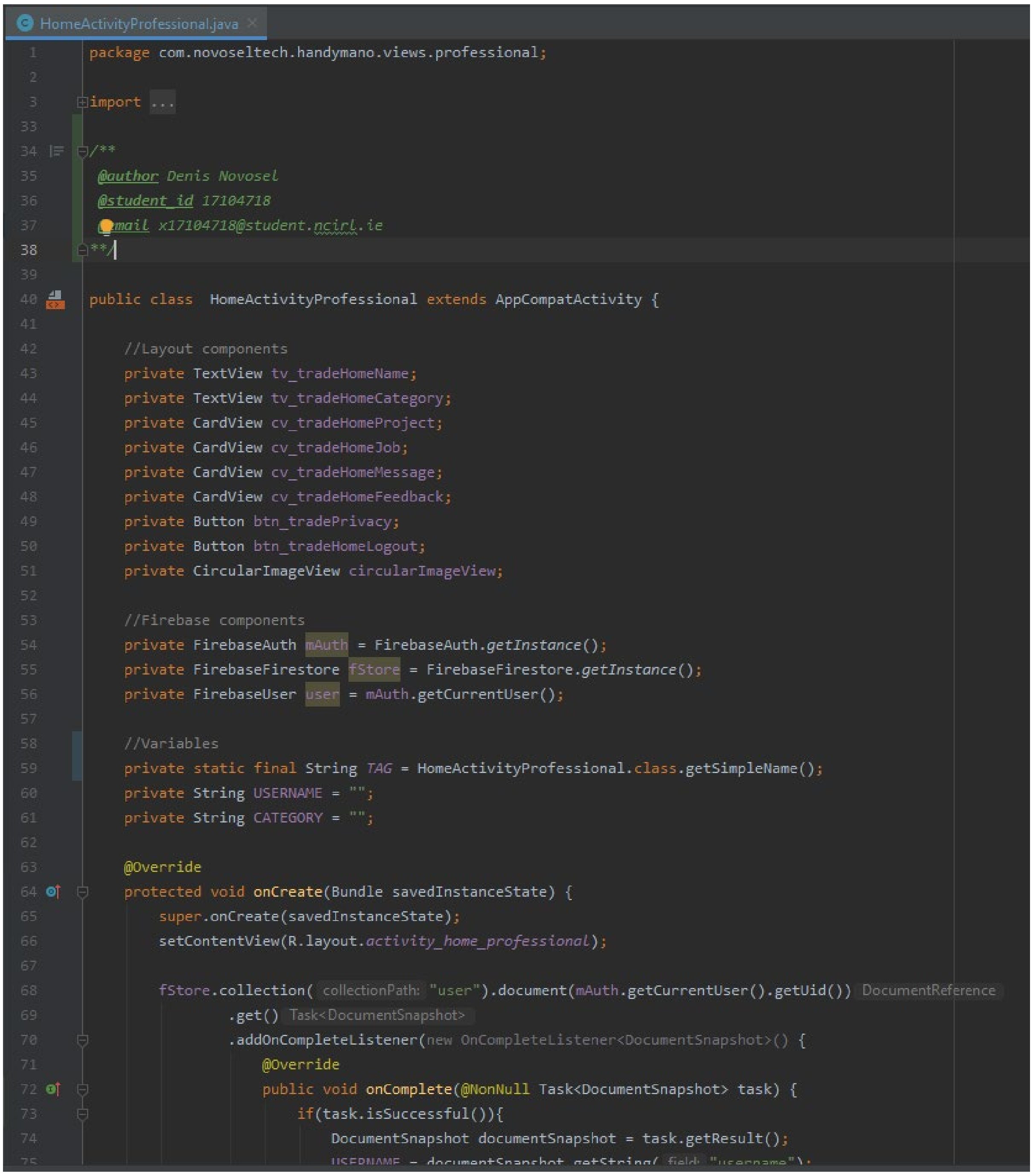
Variables and objects are mostly written as a camel case while constants are written as all capital letters.

Layout components are named with the following prefixes:

- **TextView** – *tv_*
- **TextInputLayout** – *til_*
- **EditText** – *et_*
- **ImageView** – *iv_*
- **/CircularImageView** – *civ_*
- **Button** – *btn_*
- **RecyclerView** – *rv_*
- **ScrollView** – *sv_*
- **CardView** – *cv_*
- **FrameLayout** – *fl_*
- **ConstraintLayout** – *cl_*
- **LinearLayout** – *ll_*
- **Switch** – *sw_*

2.3.2. Variable and object declaration

In the code I have consistently declared all my variables, objects and layout components globally with Private access modifier so that they are available throughout my class and then initialized most of them within the *onCreate()* method of the activity.



HomeActivityProfessional.java

```java
package com.novoseltech.handymano.views.professional;

import ...

/**
 @author Denis Novosel
 @student_id 17104718
 @mail x17104718@student.ncirl.ie
**/

public class  HomeActivityProfessional extends AppCompatActivity {

    //Layout components
    private TextView tv_tradeHomeName;
    private TextView tv_tradeHomeCategory;
    private CardView cv_tradeHomeProject;
    private CardView cv_tradeHomeJob;
    private CardView cv_tradeHomeMessage;
    private CardView cv_tradeHomeFeedback;
    private Button btn_tradePrivacy;
    private Button btn_tradeHomeLogout;
    private CircularImageView circularImageView;

    //Firebase components
    private FirebaseAuth mAuth = FirebaseAuth.getInstance();
    private FirebaseFirestore fStore = FirebaseFirestore.getInstance();
    private FirebaseUser user = mAuth.getCurrentUser();

    //Variables
    private static final String TAG = HomeActivityProfessional.class.getSimpleName();
    private String USERNAME = "";
    private String CATEGORY = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_professional);

        fStore.collection( collectionPath: "user").document(mAuth.getCurrentUser().getUid()) DocumentReference
                .get() Task<DocumentSnapshot>
                .addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if(task.isSuccessful()){
                            DocumentSnapshot documentSnapshot = task.getResult();
                            USERNAME = documentSnapshot.getString( field: "username");
```

Figure 62 - object declaration and naming

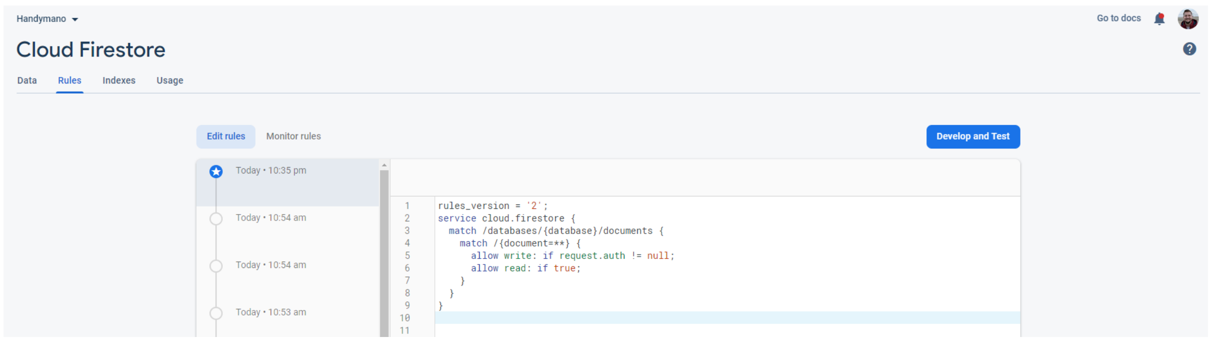
2.3.3. Firebase implementation

As it is previously specified, Handymano is using 3 modules of the Firebase which are:

- Authentication
- Firestore
- Storage

Authentication, with very suggestive name, is used for secure registration and authentication of the application users. All of the database transactions, password encryption and transport security is handled by this module. It offers various modes of registration and authentication, including social media logins, however now Handymano offers only email and password authentication.
All data validation is handled by the application before invoking the registration and sign in functionality of the Authentication module.

Firestore is NoSQL flexible and scalable database that I had previous experience with. Data operations with the database module are very easily implemented and I have used it across all areas of the Handymano.
Firestore utilized asynchronous calls so in the beginning it was very challenging to use it when I depended on the data retrieved from it, but I soon managed to get an even better understanding of its usage and from then on I have never had issues with it.
Firestore offers implementation of the custom rules and at the moment I am allowing write operations only to the authenticated users and reading of the data to all the users. Reading of the data must be allowed to all the users during the registration to check if the username and phone number are already in use by some other user.



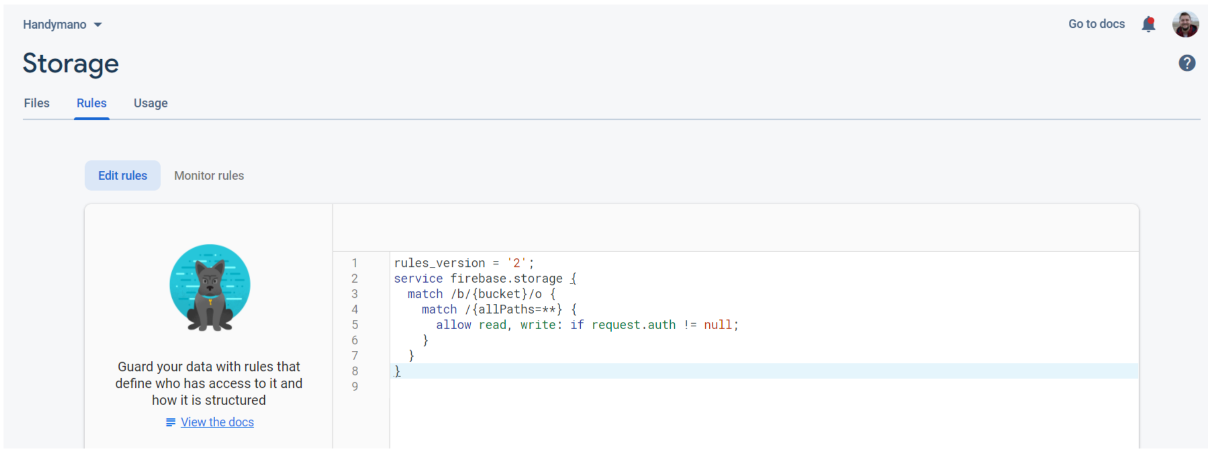
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow write: if request.auth != null;
      allow read: if true;
    }
  }
}
```

Figure 63 - Firebase Firestore rule set

Structure of Handymano Firestore database is:

```
+---myhanyman-ncirl
|       +---chat
|       |       +---{USER_ID}
|       |       |       +---{RECIPIENT_ID}
|       |       |       |       +---{MESSAGE_ID}
|         |         |         +---{recipients}
|         +---rating
|         |         +---{USER_ID}
|         |         |         +---feedback
|         |         |         |         +---{REVIEWER_ID}
|         |         |         |         |         +---{creation_date}
|         |         |         |         |         +---{feedback_text}
|         |         |         |         |         +---{stars}
|         |         |         |         |         +---{user_id}
|         |         |         |         |         +---{username}
|         +---user
|         |         +---{USER_ID}(Standard Account)
|         |         |         +---{accountType}
|         |         |         +---{email}
|         |         |         +---{phoneNo}
|         |         |         +---{username}
|         |         |         +---jobs
|         |         |         |         +---{JOB_ID}
|         |         |         |         |         +---{category}
|         |         |         |         |         +---{creation_date}
|         |         |         |         |         +---{description}
|         |         |         |         |         +---{imageCount}
|         |         |         |         |         +---{location}
|         |         |         |         |         +---{status}
|         |         |         |         |         +---{title}
|         +---user
|         |         +---{USER_ID}(Professional Account)
|         |         |         +---{accountType}
|         |         |         +---{email}
|         |         |         +---{phoneNo}
|         |         |         +---{username}
|         |         |         +---{category}
|         |         |         +---{email_visible}
|         |         |         +---{phone_visible}
|         |         |         +---{location}
|         |         |         +---{radius}
|         |         |         +---projects
|         |         |         |         +---{PROJECT_ID}
|         |         |         |         |         +---{creation_date}
|         |         |         |         |         +---{description}
|         |         |         |         |         +---{imageCount}
|         |         |         |         |         +---{title}
```

Storage is the last module of the Firebase that Handymano is using and so far there have been no issues once I got familiar with the module documentation.
Application is storing image files in the folders named after the folder creator user ID. Each folder contains optional profile image of the user and subfolders "*jobs*" or "*projects*" – depending on the user's account type. Inside of those subfolders there is another set of subfolders named after job or project title which contains images attached to the job or project.
Firebase Storage also allows rule editing, but for now I only allow Storage operations only if the user is authenticated within Handymano.


Handymano
Go to docs
Storage
Files
Rules
Usage
Edit rules
Monitor rules
Guard your data with rules that define who has access to it and how it is structured
View the docs

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Figure 64 - Firebase Storage rule set

2.3.4. Google Cloud implementation

Last part of the cloud operations is Maps SDK for Android. This is used for getting the locations of the tradespeople and jobs.
It is implemented by importing the dependencies in the Gradle build file with the API key being specified in the Manifest file.

API key is enabled only for the ***com.novoseltech.handymano*** application package as well as for the application's SHA-1 fingerprint.

2.4. Testing

Having experience in the Computer System Validation by designing and executing Installation Qualification and Operational Qualification I have decided to test a few scenarios designed below. These tests are meant to demonstrate knowledge about the developed code and prove that the application meets requirements specified in section ***2.1***.
Ideally these tests would have been executed on some Application Lifecycle Management (**ALM**) but for the purposes of this report these tests will as well demonstrate the testing process for Handymano application.

2.4.1. Account type independent testing

Standard user registration test

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that the user is logged out and that the current screen is login screen.	User is logged out and the login screen is visible.	User is logged out and the login screen is visible.
2	Click on the "Register button". Make sure registration choice activity is visible.	Registration choice activity is visible.	Registration choice activity is visible.
3	Click on "I AM LOOKING FOR A SERVICE" button. Make sure that the registration screen with 4 input fields and 2 buttons is visible.	Registration screen with 4 input fields and 2 buttons is visible.	Registration screen with 4 input fields and 2 buttons is visible.
4	Enter the following information: *Username* *Valid email* *Phone number* *Password* When all the data is entered click on "REGISTER".	User is successfully registered and automatically logged into the application.	User is successfully registered and automatically logged into the application.
5	Log out of the application and log back in with the newly created user to confirm that the login works.	After the log out user can log back in with newly created username and password.	After the log out user can log back in with newly created username and password.

Test run #1 result: PASS

Professional user registration test

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that the user is logged out and that the current screen is login screen.	User is logged out and the login screen is visible.	User is logged out and the login screen is visible.
2	Click on the "Register button". Make sure registration choice activity is visible.	Registration choice activity is visible.	Registration choice activity is visible.
3	Click on "I AM OFFERING A SERVICE" button. Make sure that the registration screen with 6 input fields and 2 buttons is visible.	Registration screen with 6 input fields and 2 buttons is visible.	Registration screen with 6 input fields and 2 buttons is visible.
4	Enter the following information: *Username* *Valid email* *Phone number* *Services category* *Years of experience* *Password* After the information is entered click on "CHOOSE LOCATION". Enter the address within Ireland and optionally enter custom traveling radius. When the data is entered click on "SEARCH" button and the pin will be added to specified address on the map. Click on "SAVE LOCATION" and you will be transferred back to the registration form seen previously. When all the data is entered click on "REGISTER".	User is successfully registered and automatically logged into the application.	User is successfully registered and automatically logged into the application.

5	Log out of the application and log back in with the newly created user to confirm that the login works.	After the log out user can log back in with newly created username and password.	After the log out user can log back in with newly created username and password.

Test run #1 result: PASS

2.4.2. Professional account type specific testing

Project create

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Professional account type.	User is logged in as Professional account.	User is logged in as Professional account.
2	On the home activity on the Professional account click on the “PROJECTS” card.	User is transferred to the activity which contains list of previously created projects.	User is transferred to the activity which contains list of previously created projects.
3	Click on the “NEW PROJECT” button at the bottom of the screen.	User is transferred to the activity which contains form to create new project.	User is transferred to the activity which contains form to create new project.
4	Enter the following information: Project title Project description Add at least 1 image Click on the “SAVE PROJECT” button.	Project is successfully created and user is transferred back to the activity which contains the list of the previously created projects. Newly created project appears on the list.	Project is successfully created and user is transferred back to the activity which contains the list of the previously created projects. Newly created project appears on the list.
5	When the project is created you are transferred to the list of created projects. Click on the newly created project.	User is transferred to the activity which contains details about selected project.	User is transferred to the activity which contains details about selected project.
6	Confirm that the data and images are those that were specified in the step 4	Data loaded is which was specified in the step 4.	Data loaded is which was specified in the step 4.

Test run #1 result: PASS

Project edit

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Professional account type.	User is logged in as Professional account.	User is logged in as Professional account.
2	On the home activity on the Professional account click on the “PROJECTS” card.	User is transferred to the activity which contains list of previously created projects.	User is transferred to the activity which contains list of previously created projects.
3	Click on one of the projects from the list.	User is transferred to the activity which displays project information.	User is transferred to the activity which displays project information.
4	In the top right corner click on the button which expands the possible actions. Select “Edit”.	User is transferred to the form which allows them to edit the project.	User is transferred to the form which allows them to edit the project.
5	In the project edit form do the following: 1. Change the project title 2. Add some text to the description 3. Delete the original image and attach new image. When complete click on “SAVE CHANGES”.	Project is saved and user is transferred to the list which contains previously created projects.	Project is saved and user is transferred to the list which contains previously created projects.
6	Open the recently edited project and confirm that the data displayed matches data specified in the step 5.	User is transferred to the activity which displays project information specified in the step 5.	User is transferred to the activity which displays project information specified in the step 5.

Test run #1 result: PASS

Project delete

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Professional account type.	User is logged in as Professional account.	User is logged in as Professional account.
2	On the home activity on the Professional account click on the “PROJECTS” card.	User is transferred to the activity which contains list of previously created projects.	User is transferred to the activity which contains list of previously created projects.
3	Click on one of the projects from the list.	User is transferred to the activity which displays project information.	User is transferred to the activity which displays project information.
4	In the top right corner click on the button which expands the possible actions. Select “Delete”. When asked to confirm is you want to delete the project select “YES”.	User is transferred to the activity which contains list of previously created projects.	User is transferred to the activity which contains list of previously created projects.
5	Confirm that the project is not available on the list of previously created projects.	Project is deleted and no longer available on the list of previously created projects.	Project is deleted and no longer available on the list of previously created projects.

Test run #1 result: *PASS*

Job listing

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that you are logged in as the Professional account type.	User is logged in as Professional account.	User is logged in as Professional account.
2	On the home activity on the Professional account click on the “JOBS” card.	User is transferred to the activity which contains list of jobs available within the radius of travel and category of services provided.	User is transferred to the activity which contains list of jobs available within the radius of travel and category of services provided.
3	Click on one of the jobs from the list.	User is transferred to the activity which displays job information.	User is transferred to the activity which displays job information.
4	Confirm that the following data is visible: 1. Job ad title 2. Job posting information 3. Job ad address 4. Job image(s) 5. Job description	All the job ad related data is displayed on the activity.	All the job ad related data is displayed on the activity.

Test run #1 result: PASS

Job advertiser messaging

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that you are logged in as the Professional account type.	User is logged in as Professional account.	User is logged in as Professional account.
2	On the home activity on the Professional account click on the “JOBS” card.	User is transferred to the activity which contains list of jobs available within the radius of travel and category of services provided.	User is transferred to the activity which contains list of jobs available within the radius of travel and category of services provided.
3	Click on one of the jobs from the list.	User is transferred to the activity which displays job information.	User is transferred to the activity which displays job information.
4	Click on “MESSAGE ADVERTISER” button.	Chat window is open with the job advertiser and the default greeting message is loading in the message input field.	Chat window is open with the job advertiser and the default greeting message is loading in the message input field.
5	Keep the default message or specify your own. Click on icon to send the message.	Message is sent and the message appears in the chat window within the message bubble.	Message is sent and the message appears in the chat window within the message bubble.
6	Hit the “Back” button on your device twice to go back to the job ads list. Click on the top left corner “burger” icon to expand the navigation drawer. Select “Messages”.	User is transferred to the message menu activity.	User is transferred to the message menu activity.
7	Confirm that there is an entry available in the list of messages that was recently sent.	Message entry with job poster display name is available in the list.	Message entry with job poster display name is available in the list.

Test run #1 result: PASS

2.4.3. Standard account type specific testing

Job ad create

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	Expand the navigation drawer by clicking "burger" icon in the top left corner. From the navigation drawer select "Jobs".	User is transferred to the activity which contains list of previously created jobs.	User is transferred to the activity which contains list of previously created jobs.
3	Click on the "NEW JOB" button at the bottom of the screen.	User is transferred to the activity which contains form to create new job.	User is transferred to the activity which contains form to create new job.
4	Enter the following information: Job title Job description Add at least 1 image Job category Job address Click on the "SAVE JOB" button.	Job is successfully created and user is transferred back to the activity which contains the list of the previously created jobs. Newly created job appears on the list.	Job is successfully created and user is transferred back to the activity which contains the list of the previously created jobs. Newly created job appears on the list.
5	When the job is created you are transferred to the list of created jobs. Click on the newly created job.	User is transferred to the activity which contains details about selected job.	User is transferred to the activity which contains details about selected job.
6	Confirm that the data and images are those that were specified in the step 4	Data loaded is which was specified in the step 4.	Data loaded is which was specified in the step 4.

Test run #1 result: PASS

Job ad edit

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	Expand the navigation drawer by clicking “burger” icon in the top left corner. From the navigation drawer select “Jobs”.	User is transferred to the activity which contains list of previously created jobs.	User is transferred to the activity which contains list of previously created jobs.
3	Click on one of the jobs from the list.	User is transferred to the activity which displays job information.	User is transferred to the activity which displays job information.
4	In the top right corner click on the button which expands the possible actions. Select “Edit”.	User is transferred to the form which allows them to edit the job.	User is transferred to the form which allows them to edit the job.
5	In the job edit form do the following: 1. Change the job title 2. Add some text to the description 3. Delete the original image and attach new image. 4. Change the job category. 5. Set the job ad status as Private. When complete click on “SAVE CHANGES”.	Job is saved and user is transferred to the list which contains previously created jobs.	Job is saved and user is transferred to the list which contains previously created jobs.
6	Open the recently edited job and confirm that the data displayed matches data specified in the step 5.	User is transferred to the activity which displays job information specified in the step 5.	User is transferred to the activity which displays job information specified in the step 5.

Test run #1 result: PASS

Job ad delete

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1.	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	Expand the navigation drawer by clicking “burger” icon in the top left corner. From the navigation drawer select “Jobs”.	User is transferred to the activity which contains list of previously created jobs.	User is transferred to the activity which contains list of previously created jobs.
3	Click on one of the jobs from the list.	User is transferred to the activity which displays job information.	User is transferred to the activity which displays job information.
4	In the top right corner click on the button which expands the possible actions. Select “Delete”. When asked to confirm is you want to delete the job select “YES”.	User is transferred to the activity which contains list of previously created jobs.	User is transferred to the activity which contains list of previously created jobs.
5	Confirm that the project is not available on the list of previously created projects.	Job is deleted and no longer available on the list of previously created jobs.	Job is deleted and no longer available on the list of previously created jobs.

Test run #1 result: PASS

Browse for tradespeople

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	On the home activity leave the option “Current location”, “Category of services” as “All categories” and click on search.	Entries with the users which match the criteria of the search are returned and displayed in the list.	Entries with the users which match the criteria of the search are returned and displayed in the list.
3	Click on the “EXPAND SEARCH”. Choose specific category of services and click again on “SEARCH”.	User(s) matching the category and location criteria of the search are returned and displayed in the list.	User(s) matching the category and location criteria of the search are returned and displayed in the list.
4.	Click on the “EXPAND SEARCH” again. Click on the “location” icon to change the search mode from “Current location” to “Custom address”. Specify the address and click again on “SEARCH”.	User(s) matching the category and location criteria of the search are returned and displayed in the list.	User(s) matching the category and location criteria of the search are returned and displayed in the list.

Test run #1 result: PASS

Add rating and feedback

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	On the home activity leave the option “Current location”, “Category of services” as “All categories” and click on search.	Entries with the users which match the criteria of the search are returned and displayed in the list.	Entries with the users which match the criteria of the search are returned and displayed in the list.
3	From the list of returned users click on one of them.	User is transferred to the profile activity of the professional user.	User is transferred to the profile activity of the professional user.
4	Click on “FEEDBACK” button.	User is transferred to the feedback and rating	User is transferred to the feedback and

		activity for that Professional user.	rating activity for that Professional user.
5	Click on “ADD FEEDBACK” button to rate user and leave feedback.	User is transferred to the form where they can rate the user and write optional feedback.	User is transferred to the form where they can rate the user and write optional feedback.
6	Select the rating 1-5 stars and write some comment. Click on “SAVE FEEDBACK”.	Feedback is saved and user is transferred back to the profile of tradesman.	Feedback is saved and user is transferred back to the profile of tradesman.
7	Click again on “FEEDBACK” button. Confirm that your feedback entry is visible on the feedback list and that button states “EDIT FEEDBACK”.	Feedback previously created is visible on the list. Button which previously stated “ADD FEEDBACK” now states “EDIT FEEDBACK”.	Feedback previously created is visible on the list. Button which previously stated “ADD FEEDBACK” now states “EDIT FEEDBACK”.
8	Click on “EDIT FEEDBACK”. When transferred to the feedback editing form, delete the feedback comment. Click on “SAVE FEEDBACK”.	Feedback is saved and user is transferred back to the profile of tradesman.	Feedback is saved and user is transferred back to the profile of tradesman.
9	Click again on “FEEDBACK” button. Confirm that your feedback entry is not visible on the feedback list, but the rating is still accounted for and that button states “EDIT FEEDBACK”.	Feedback previously created is not visible on the list. Rating is still accounted for. Button which previously stated “ADD FEEDBACK” now states “EDIT FEEDBACK”.	Feedback previously created is not visible on the list. Rating is still accounted for. Button which previously stated “ADD FEEDBACK” now states “EDIT FEEDBACK”.
10	Add feedback comment again and save the feedback. Confirm that the feedback is again visible on the feedback list.	Feedback was saved. Feedback is again visible in the feedback list.	Feedback was saved. Feedback is again visible in the feedback list.
11	Hold the click on your feedback entry. You will be prompted to delete the feedback. Select “YES”. You are transferred to the profile of the tradesman.	Feedback is removed from the list of user’s feedback and the rating is no longer accounted.	Feedback is removed from the list of user’s feedback and the rating is no longer accounted.

	Click again on the "FEEDBACK" button and confirm that the feedback is not visible in the list as well as that the rating is no longer counted.		

Test run #1 result: PASS

Message tradesman

Test run #1

STEP	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
1	Open the application. Make sure that you are logged in as the Standard account type.	User is logged in as Standard account.	User is logged in as Standard account.
2	On the home activity leave the option "Current location", "Category of services" as "All categories" and click on search.	Entries with the users which match the criteria of the search are returned and displayed in the list.	Entries with the users which match the criteria of the search are returned and displayed in the list.
3	From the list of returned users click on one of them.	User is transferred to the profile activity of the professional user.	User is transferred to the profile activity of the professional user.
4	Click on the message icon next to the user's name.	Chat window with the tradesman is open.	Chat window with the tradesman is open.
5	Type some message and click on the send message icon.	Message is sent and the message appears in the chat window within the message bubble.	Message is sent and the message appears in the chat window within the message bubble.
6	Hit the "Back" button on your device to go back to the profile of the tradesman. Click on the top left corner "burger" icon to expand the navigation drawer. Select "Messages".	User is transferred to the message menu activity.	User is transferred to the message menu activity.
7	Confirm that there is an entry available in the list of messages that was recently sent.	Message entry with tradesman's display name is available in the list.	Message entry with tradesman's display name is available in the list.

Test run #1 result: PASS

3. Conclusions

I really enjoyed working on this project and found development of Android based applications very interesting. Throughout the whole lifecycle of this project I was challenged many time juggling both work and last year of college, moved to Switzerland and changed the job and all of that while there is a pandemic happening in the world.

I am very proud of what I have achieved in this project with all of the above circumstances, even though in my mind I had so much more planned but I was constrained by the time period and resources.

As of now I would not want this application to be used commercially until all of my ideas are implemented and then it will be decided.

But what I am most satisfied with is how much I learned about development for the Android platform. I actually proved it to myself that I can learn by doing and no amount of studying beats wrapping up your sleeves and just developing, failing, learning on your mistakes and powering through it.

Right now I am on the level where I can develop so much functionality without looking for the help online and this will surely help me in the further stages of my personal and professional life.

4. Further Development

I have given a lot of though whether I want to continue developing this system or not because after 4 years of college I want to take a little break. But with this break I can actually commit to learn Spring Boot as I planned to do this year and learn it by developing web application version of Handymano.

I am not yet sure whether I will continue and publish Handymano as an available application or just use it for the learning and experience purposes but surely I will still develop it.

From the features I plan to do in the future is using Cloud Vision API from Google Cloud to detect any inappropriate content in the images attached and building web application that can be accessed from the browsers which would have an administration module instead of administering data directly in the Firebase.

5. References

Android Developers. *Documentation | Android Developers*. [online] Available at: <https://developer.android.com/docs>.

Firebase. *Documentation | Firebase*. [online] Available at: <https://firebase.google.com/docs>.

Google Developers. *Maps SDK for Android overview | Google Developers*. [online] Available at: <https://developers.google.com/maps/documentation/android-sdk/overview>.

Google Codelabs. *Google Codelabs*. [online] Available at: <https://codelabs.developers.google.com/>.

Material Design. *Material Design*. [online] Available at: <https://www.material.io/>.

GitHub. *firebase/FirebaseUI-Android*. [online] Available at: <https://github.com/firebase/FirebaseUI-Android>.

GitHub. *smarteist/Android-Image-Slider*. [online] Available at: <https://github.com/smarteist/Android-Image-Slider>.

GitHub. *ocpsoft/prettytime*. [online] Available at: <https://github.com/ocpsoft/prettytime>.

GitHub. *bumptech/glide*. [online] Available at: <https://github.com/bumptech/glide>.

Codebrainer.com. *RecyclerView for Android Beginners - How to display data*. [online] Available at: <https://www.codebrainer.com/blog/how-to-display-data-with-recyclerview>.

Medium. *Simple Android Chat Application using FirestoreRecyclerAdapter*. [online] Available at: <https://medium.com/@akhilkc9/simple-android-chat-application-using-firestorerecycleradapter-7f632da2eaee>.

Stack Overflow. 2017. *How to Convert Image URL to File in android*. [online] Available at: <https://stackoverflow.com/questions/46440777/how-to-convert-image-url-to-file-in-android>.

Stack Overflow. 2012. *How to call a method after a delay in Android*. [online] Available at: <https://stackoverflow.com/questions/3072173/how-to-call-a-method-after-a-delay-in-android>.

Stack Overflow. 2010. *How to get Bitmap from an Uri?*. [online] Available at: <https://stackoverflow.com/questions/3879992/how-to-get-bitmap-from-an-uri>.

Stack Overflow. 2019. *How to save an image in a subdirectory on android Q whilst remaining backwards compatible*. [online] Available at: <https://stackoverflow.com/questions/57030990/how-to-save-an-image-in-a-subdirectory-on-android-q-whilst-remaining-backwards-c>.

Stack Overflow. 2014. *RecyclerView onClick*. [online] Available at: <https://stackoverflow.com/questions/24471109/recyclerview-onclick/26196831#26196831>.

Stack Overflow. 2012. *How to get complete address from latitude and longitude?*. [online] Available at: <https://stackoverflow.com/questions/9409195/how-to-get-complete-address-from-latitude-and-longitude>.

TVAC Studio, 2020. *Firestore Database to RecyclerView - Android Studio Tutorial*. [video] Available at: <https://www.youtube.com/watch?v=cBwaJYocb9I>.

6. Appendices

6.1. Project Proposal

6.1.1. Objectives

Objective of my application is to develop a platform that will allow businesses and tradesmen to register their account, specify services they are offering, create portfolio of previously completed projects, receive feedback from users and respond to job ads.
Another objective of this platform is to accept regular users, allow them to search for registered trades and tradesmen at the location they specify and post job advertisements if they require some work to be completed.
Last objective is to create the functionality so that professional and regular accounts can communicate with each other such as tradesmen responding to an ad or regular user querying the tradesman or business about the service(s) they offer.

6.1.2. Background

Recently I have spent some time on the freelance websites for software developers, but also during that time I required a plumber to fix the issue in my apartment. Because I did not want to blindly hire someone to do this for me it meant I had to spend some time looking for a person with desired level of experience. I spent significant amount of time browsing through Facebook groups, Google Maps and forums until I decided on whom to hire. There simply had to be a better and more efficient way to find reliable and already reviewed tradesmen.

That is when I decided that I wish to build a platform that would allow me to search for reputable trades based on my location and my needs.

The idea seemed fantastic to me, but I also had a feeling I can expand it even further without “over-engineering” it. That is why I have decided that I wish to add advertisement board into the equation.
I would really want to post an ad where I need some physical activity completed within some time frame and that registered tradesmen and business can respond to my ad and offer their services without me looking for them.

6.1.3. Technical Approach

I will first work on creating the initial designs of my application using Adobe XD. By using I will have a guide of what I wish to build and how I want and need it to look instead of doing the designs straight from the head.

Next step is to create repository on GitHub. I need to do this so that I can always have my project safely stored and I can easily rollback it in case there is some major issue with the platform. Using GitHub will also be useful when I wish to share the current project with my supervisor.

After code repository is ready I need to prepare the data storage. For this I will need to create my project of Google Firebase and set up all the policies to allow for successful data storage and retrieval.

Following completion of designs, code repository and data storage creation I can start developing code. First core functionality I will develop will be registration and login of personal and professional accounts.

Next core functionality to develop will be navigation drawer to be able to navigate across the application. Navigation drawer will be different based on the account type.

Once I have the navigation drawer complete I will create most of the remaining layouts of the application and link navigation between them.

After navigation throughout application and layouts are completed I will develop remainder of the functionality.

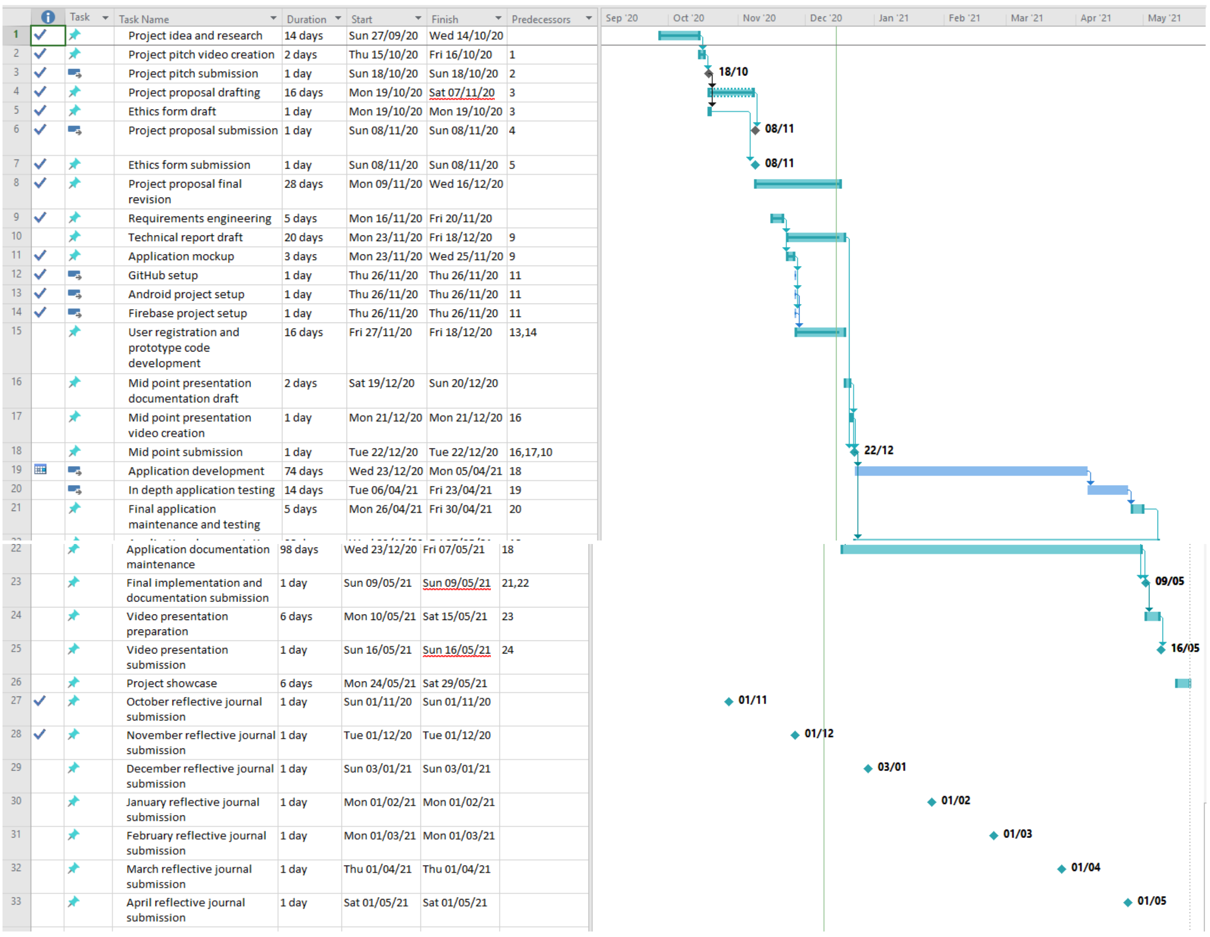
6.1.4. Technical Details

Throughout this project I will use the following technologies:

1. **Android SDK** – the entire application will be developed and deployed on Google's Android platform. Even though I recently started learning about Android application development I will be using Java language to code application logic. I do not believe I will run into any major issues, especially with the extensive number of tutorials and guides offered online.
2. **Google Firebase** – it will be used to store all of the application's data. I decided to use Google Firebase as my data storage because of it's simplicity to use, large number of innovative features on the free plan and very easy integration with Android SDK.
3. **Google Maps API** – this API will be used to determine location of the user and proximity to the registered trades in Ireland whose willingness to travel is not set to all of Ireland
4. **Picasso** – it is powerful image manipulation and caching library for Android. I will utilize it to handle image files before their submission to the Firebase storage as well as for displaying them for user.
5. **GitHub** – GitHub will be my main workspace to have my code properly version controlled as well as securely stored. For each new feature that is developed I will create new branch.
6. **Adobe XD** – I will use this tool to develop mock-ups of my application and have them as the reference design.

6.1.5. Project Plan

I have created new Microsoft Project and developed Gantt chart to lay out what in my opinion are most important tasks and milestones. As it is with many projects, dates might change but I will be able to track and prioritize tasks by having Gantt chart as my main time management hub.



		Task	Task Name	Duration	Start	Finish	Predecessors
1	✓		Project idea and research	14 days	Sun 27/09/20	Wed 14/10/20	
2	✓		Project pitch video creation	2 days	Thu 15/10/20	Fri 16/10/20	1
3	✓		Project pitch submission	1 day	Sun 18/10/20	Sun 18/10/20	2
4	✓		Project proposal drafting	16 days	Mon 19/10/20	Sat 07/11/20	3
5	✓		Ethics form draft	1 day	Mon 19/10/20	Mon 19/10/20	3
6	✓		Project proposal submission	1 day	Sun 08/11/20	Sun 08/11/20	4
7	✓		Ethics form submission	1 day	Sun 08/11/20	Sun 08/11/20	5
8	✓		Project proposal final revision	28 days	Mon 09/11/20	Wed 16/12/20	
9	✓		Requirements engineering	5 days	Mon 16/11/20	Fri 20/11/20	
10			Technical report draft	20 days	Mon 23/11/20	Fri 18/12/20	9
11	✓		Application mockup	3 days	Mon 23/11/20	Wed 25/11/20	9
12	✓		GitHub setup	1 day	Thu 26/11/20	Thu 26/11/20	11
13	✓		Android project setup	1 day	Thu 26/11/20	Thu 26/11/20	11
14	✓		Firebase project setup	1 day	Thu 26/11/20	Thu 26/11/20	11
15			User registration and prototype code development	16 days	Fri 27/11/20	Fri 18/12/20	13,14
16			Mid point presentation documentation draft	2 days	Sat 19/12/20	Sun 20/12/20	
17			Mid point presentation video creation	1 day	Mon 21/12/20	Mon 21/12/20	16
18			Mid point submission	1 day	Tue 22/12/20	Tue 22/12/20	16,17,10
19			Application development	74 days	Wed 23/12/20	Mon 05/04/21	18
20			In depth application testing	14 days	Tue 06/04/21	Fri 23/04/21	19
21			Final application maintenance and testing	5 days	Mon 26/04/21	Fri 30/04/21	20
22			Application documentation maintenance	98 days	Wed 23/12/20	Fri 07/05/21	18
23			Final implementation and documentation submission	1 day	Sun 09/05/21	Sun 09/05/21	21,22
24			Video presentation preparation	6 days	Mon 10/05/21	Sat 15/05/21	23
25			Video presentation submission	1 day	Sun 16/05/21	Sun 16/05/21	24
26			Project showcase	6 days	Mon 24/05/21	Sat 29/05/21	
27	✓		October reflective journal submission	1 day	Sun 01/11/20	Sun 01/11/20	
28	✓		November reflective journal submission	1 day	Tue 01/12/20	Tue 01/12/20	
29			December reflective journal submission	1 day	Sun 03/01/21	Sun 03/01/21	
30			January reflective journal submission	1 day	Mon 01/02/21	Mon 01/02/21	
31			February reflective journal submission	1 day	Mon 01/03/21	Mon 01/03/21	
32			March reflective journal submission	1 day	Thu 01/04/21	Thu 01/04/21	
33			April reflective journal submission	1 day	Sat 01/05/21	Sat 01/05/21	

6.1.6. Evaluation

Testing my application is as equally important to me as developing quality code to accommodate for required functionality.

For the evaluation of my application I will perform automated user interface tests using Espresso framework because of it's stability, speed and effortless integration with Android Studio.

I will also provide the application to some of my friends who are not engineering-oriented to obtain insights from end user's point of view. By receiving their feedback, I will do my best to make application as user friendly as possible.

6.1.7. References

1. Android Developers. n.d. [online] Available at: <https://developer.android.com/>
2. Firebase. n.d. *Firebase*. [online] Available at: <https://firebase.google.com/>.

3. Google Developers. n.d. *Google Maps Platform | Google Developers*. [online] Available at: <https://developers.google.com/maps/documentation>.
4. Square.github.io. n.d. *Picasso*. [online] Available at: <https://square.github.io/picasso/>.
5. GitHub. n.d. *Github: Where The World Builds Software*. [online] Available at: <https://github.com/>.
6. Adobe. n.d. *Adobe XD | Fast & Powerful UI/UX Design & Collaboration Tool*. [online] Available at: <https://www.adobe.com/ie/products/xd.html>.

6.2. Reflective Journals

6.2.1. October reflective journal

12 October 2020

On Saturday we have another "Software Project" session and we were introduced to the concept of Reflective journal. Under today's entry I will go over all my thoughts so far.

Going throughout this summer I had plenty of thoughts as to what should I work on as my final project. My main idea was to create the mobile fitness application which would allow users to track their workout, nutrition and measurements. It would also use Google Maps API and would allow them to browse for CERTIFIED fitness trainers in the area they select and enrol into their program. Application would also support payment so I was thinking I could use Stripe as payment gateway.
Results of the client would be visible to the trainer in their dedicated application.
I gave up on this idea because in August I hired a good friend of mine as my coach and the application he uses with me does all of this except for map based search.
I decided I will use parts of this idea for the project in "Multimedia and Mobile application development" module.

Next idea I was sure will be my project was to use OPC UA SDK to be able to connect to OPC UA servers which are connected to PLCs that are providing real-time manufacturing data via tags and store it in the database. That data could then be modelled into dashboards and used on the go via mobile application. My manager was so thrilled when I told him about this idea.
However, I will have to let him down since there are already products such as this one on the market and they are developed by the companies with thousands of expert developers. I myself am no match to beat that.

I still have few days to think about what to do. Trait about me I am very proud of is that I do not panic as deadline is getting closer. Instead I keep cool head and things always work out as they should.

17 October 2020

I have submitted my video pitch today. Idea came to me about an Android application that would use Google Maps API and allow users to look for the registered tradesmen in the specified area based on the category of the work they need performed. Application would return results on scrollable activity. Registered tradesmen would have an option to specify how much are they willing to travel.

Users would also be able to post ads for the type of work they need made and it would incorporate “freelancer” concept. For example, I need garden table and two benches made and registered carpenter can offer their services. I can visit their profile, their portfolio and if I am confident in them I can hire them.
I hope NCI accepts my idea as I have not found anything that is the same as this.

28 October 2020

I see that the list of supervisors has been published, but we still have not received any feedback about our projects. Waiting for the feedback and to be honest, can’t wait to start working on the project despite the workload from other projects.

6.2.2. November reflective journal

12 October 2020

On Saturday we have another “Software Project” session and we were introduced to the concept of Reflective journal. Under today’s entry I will go over all my thoughts so far.

Going throughout this summer I had plenty of thoughts as to what should I work on as my final project. My main idea was to create the mobile fitness application which would allow users to track their workout, nutrition and measurements. It would also use Google Maps API and would allow them to browse for CERTIFIED fitness trainers in the area they select and enrol into their program. Application would also support payment so I was thinking I could use Stripe as payment gateway.
Results of the client would be visible to the trainer in their dedicated application.
I gave up on this idea because in August I hired a good friend of mine as my coach and the application he uses with me does all of this except for map based search.
I decided I will use parts of this idea for the project in “Multimedia and Mobile application development” module.

Next idea I was sure will be my project was to use OPC UA SDK to be able to connect to OPC UA servers which are connected to PLCs that are providing real-time manufacturing data via tags and store it in the database. That data could then be modelled into dashboards and used on the go via mobile application. My manager was so thrilled when I told him about this idea.
However, I will have to let him down since there are already products such as this one on the market and they are developed by the companies with thousands of expert developers. I myself am no match to beat that.

I still have few days to think about what to do. Trait about me I am very proud of is that I do not panic as deadline is getting closer. Instead I keep cool head and things always work out as they should.

17 October 2020

I have submitted my video pitch today. Idea came to me about an Android application that would use Google Maps API and allow users to look for the registered tradesmen in the specified area based on the category of the work they need performed. Application would return results on scrollable activity. Registered tradesmen would have an option to specify how much are they willing to travel.

Users would also be able to post ads for the type of work they need made and it would incorporate "freelancer" concept. For example, I need garden table and two benches made and registered carpenter can offer their services. I can visit their profile, their portfolio and if I am confident in them I can hire them.
I hope NCI accepts my idea as I have not found anything that is the same as this.

28 October 2020

I see that the list of supervisors has been published, but we still have not received any feedback about our projects. Waiting for the feedback and to be honest, can't wait to start working on the project despite the workload from other projects.

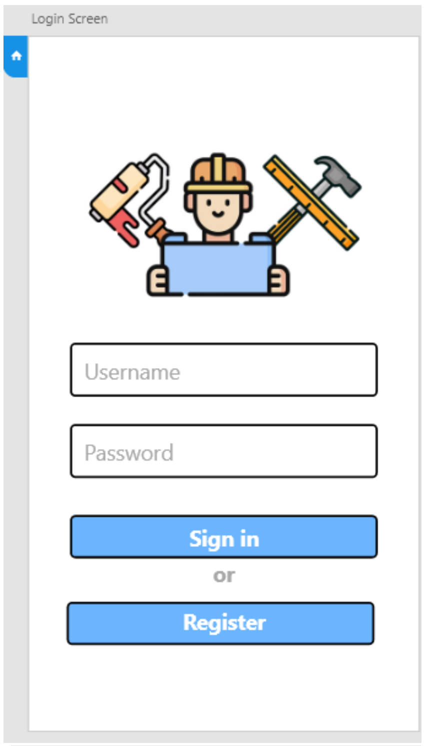
4 November 2020

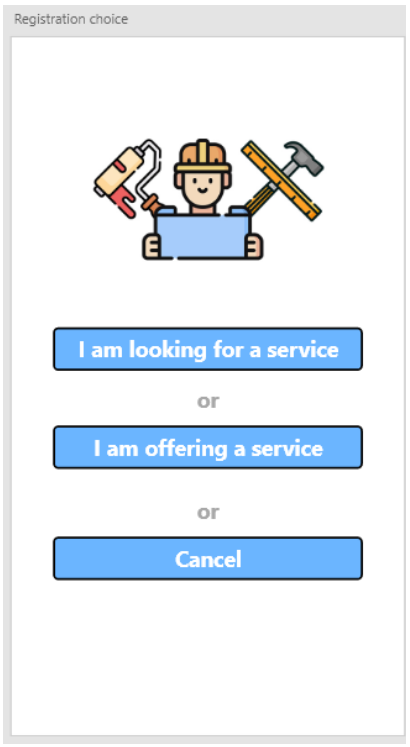
After getting in touch with my supervisor I got the feedback on my project where I was notified that I can go ahead with my proposed idea without any alterations.

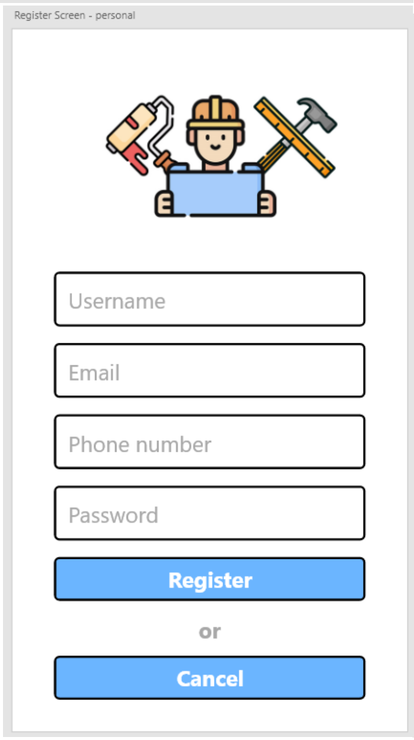
Currently I am busy working on various modules and I want to complete some workload prior to adding another task to my schedule.

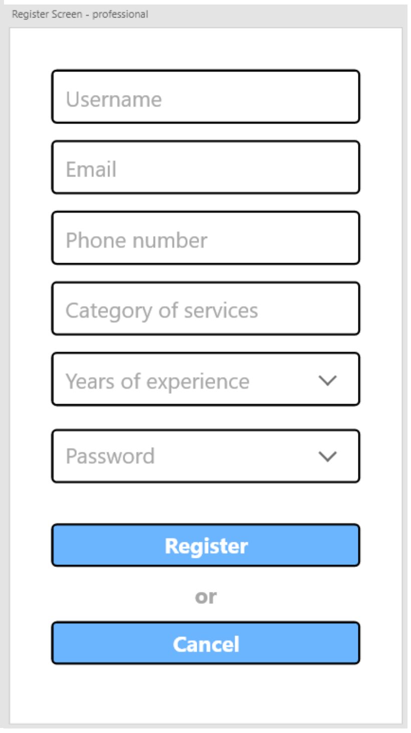
20 November 2020

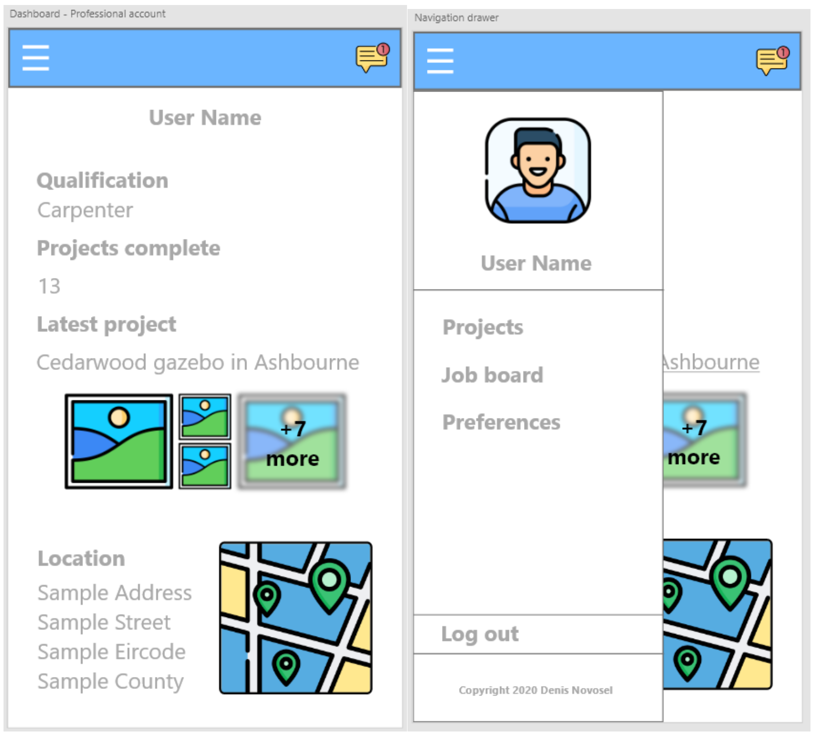
I have started with the design for some of the activities of the Android application. They are only the sketches and are not representation of the final product.


Login Screen
Username
Password
Sign in
or
Register


Registration choice
I am looking for a service
or
I am offering a service
or
Cancel


Register Screen - personal
Username
Email
Phone number
Password
Register
or
Cancel


Register Screen - professional
Username
Email
Phone number
Category of services
Years of experience
Password
Register
or
Cancel



23 November 2020

I have created the project in Android Studio and started with designing of initial activities. For the graphic choice I will be using a mix of Material Design for components and free icons from flaticon.com for the graphics.

26 November 2020

I have decided to go with Google Firebase as my data storage provider. I am using Firebase in a project for Multimedia and mobile application development and I was impressed how easy it is to achieve things such as user registration and login, session maintenance and storing data in document database.

6.2.3. December reflective journal

1 December 2020

With first of December comes first version of code to my GitHub repository. I have completed user authentication and registration system for the application using Firebase Authentication module.

4 December 2020

Today I had first meeting with my supervisor regarding project proposal and mid-point submission. We established that my project proposal requires more information added and that we would meet again in 2 weeks to go over the final revisions of mid-point documentation.

23 December 2020

This will be the last entry for month of December. Yesterday I have submitted the mid-point documentation and video presentation of my current progress. So far I have completed the user registration and login, navigation drawer system and started work on data retrieval from Firebase.

6.2.4. January reflective journal

14 January 2020

I met with my supervisor today to briefly discuss feedback on my midpoint submission. I cannot say that I am satisfied considering the amount of work I have put into both final revision of project proposal and first version of technical report. Admittedly, my prototype was not developed to the level that I was satisfied with but I surely did good job at that point.
Supervisor tasked me with creation of the list of features I wish to implement by middle of the February.

28 January 2020

I have set the following goals for implementation by middle of February:

1. **Registration and authentication system completion** – obtaining more information during user registration, expanding complexity of data validation during user registration, email confirmation of registration and password reset functionality.

2. **Location based professional services search** – connect the project with Google Cloud and Maps API, implement retrieval of professional accounts by location and category, visual customization of data retrieved.

3. **Profile customizations for both account types**

Today I met with supervisor and showed him the current progress, he agreed I am on track with my mid-February goals.

30 January 2020

This will be the last entry in journal for month of January. So far I have completed my first two mid-February goals and I am yet to start with goal #3 which I anticipate to complete by the end of week starting 1st of February.

6.2.5. February reflective journal

8 February 2021

Since the last entry in the journal I have not done a lot of work on my project. Unfortunately, I am very busy with my employment due to large number of concurrent projects we are executing and after many work hours I lose focus as soon as I start some work on college assignments.

I hope to gain the grip again and continue my work as expected.

15 February 2021

I have found some free time to get back on the track with my software project. I mostly completed my mid-February goal expect for the "professional" account profile customization. It requires work with both location selection fragment as well as spinners and lots of database updates handling and I was just too lazy unfortunately to power through it. I will leave that bit for the meeting with the supervisor.

For the mid-March goals I have the following:

1. **"Professional" account profile customization** – this activity needs to be completed and is carried over from previous short-term plan

2. **Project creation system for professional account types** – this functionality should allow "Professional" users to create individual projects under their profile. Each project will have title, description and images.

3. **Job ad creation system for "Standard" account types** – this functionality should allow "Standard" users to create job ads. These ads should contain title, description and images (optionally).

I am fairly confident that I will complete these tasks as I am now very experienced with both Android development and Firebase so the development flow is very smooth without any major setbacks.

25 February 2021

I could not meet with the supervisor yesterday for the feedback on the current progress. I am still on the track for points previously mentioned and have done significant progress regarding point #2. I did not like working with images in Android but I have got comfortable with them, especially due to the plugin named "*Glide*" which makes it very easy to load images into ImageView.

Supervisor advised me to send the updates on the email.

28 February 2021

I still haven't sent an email with the updates to the supervisor because I wanted to complete point #2 but due to heavy workload from the rest of modules I could not finish it. Most of the logic for that feature is complete and minor bug fixes and visual updates are required.
I will send the update tomorrow.

6.2.6. March reflective journal

6 March 2021

I am almost done with another project for Usability Design, but then remains another project for Cloud Application Development. There have not really been many significant developments for my project.

20 March 2021

Job and project creating, deleting and editing systems are complete. It was a bit tricky to figure out how to manage editing of currently added images due to need for re-uploading them for job and project but in the end I decided to go with downloading images into a temporary folder and after saving the images I then delete the temporary created folder.

30 March 2021

Unfortunately, I did not touch the project because I committed myself to another project which is due 2nd of April for Cloud Application Development. On the other hand I got the work permit for Switzerland and I will be flying there on 7th of April. Once I settle in Switzerland then I will continue further development on my project.

6.2.7. April reflective journal

11 April 2021

Couple of days into my move to Switzerland, I have completed some layout designing and completed browsing of jobs posted for tradesmen. Also regular users can now visit tradesman profile and see their address and projects.

16 April 2021

I have completed the messaging system and now regular users are able to message tradesmen by visiting their profile and tradesmen are able to message regular users by visiting job postings. Maybe I should also learn how to implement it as a service so that users receive message notification.

30 April 2021

I have almost completed the feedback and rating system. Only thing left are some minor design changes for this functionality and will be done over the weekend. The project deadline is near and functionality is almost fully complete. Over the course of next week I will focus on final changes, testing and documentation updates.