



# **THE VOLUNTEERING NETWORK – PLATFORM FOR CONNECTING VOLUNTEERS AND COMPANIES USING FLUTTER**

A thesis submitted to  
the Faculty of Engineering and Natural Sciences of  
International University of Sarajevo in partial  
fulfillment of the requirements for the  
degree of Bachelor of Science  
in COMPUTER SCIENCES AND ENGINEERING  
and SOFTWARE ENGINEERING

by  
Nedim Kunovac, Mirza Redžepović and Edin Žiga

June, 2023

Thesis written by

NEDIM KUNOVAC, MIRZA REDŽEPOVIĆ and EDIN ŽIGA

Graduation Committee

Assoc. Prof. Dr. Kanita Karađuzović-Hadžiabdić	International University of Sarajevo, Bosnia and Herzegovina, Supervisor
Assoc. Prof. Dr. Khaldoun Al Khalidi	International University of Sarajevo, Bosnia and Herzegovina
Assoc. Prof. Dr. Zeynep Sağır	International University of Sarajevo, Bosnia and Herzegovina
Assoc. Prof. Dr. Ali Abd Almisreb	International University of Sarajevo, Bosnia and Herzegovina

Signatures

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Date

\_\_\_\_\_

We hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. We also declare that, as required by these rules and conduct, we have fully cited and referenced all material and results that are not original to this work.

NEDIM KUNOVAC

MIRZA REDŽEPOVIĆ

EDIN ŽIGA

# INTERNATIONAL UNIVERSITY OF SARAJEVO

## DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED WORK

Copyright 2023 © by Nedim Kunovac, Mirza Redžepović, Edin Žiga rights reserved.

### THE VOLUNTEERING NETWORK – PLATFORM FOR CONNECTING VOLUNTEERS AND COMPANIES USING FLUTTER

No part of this unpublished work may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without prior written permission of the copyright holder and IUS Library.

Affirmed by Nedim Kunovac, Mirza Redžepović, Edin Žiga

Signatures

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Date

\_\_\_\_\_

## **ABSTRACT**

Due to the current unavailability of proper and skillful volunteers for groups such as non-profits and community groups, it is very hard for them to work and function properly. However, the main issue is that there is no proper mean of applying and viewing all volunteering opportunities, as well as reviewing and picking volunteers with fitted profiles. Such systems are not present today, leaving a gap between the volunteers looking for work and the companies looking for the volunteers. The main idea behind this platform is to not only create a “bridge” which covers the gap, but also to reduce the tedious processes of searching and recruiting. This thesis provides the detailed architecture and implementation of this platform, along with the explanations for each step. The project as a whole was pitched by Senior Mobile Engineer Alen Seferović, who works at the company Ant Colony, based in Sarajevo, Bosnia and Herzegovina. For development, Flutter was utilized and taken advantage of, with the purpose of creating a cross-platform mobile application which used a serverless architecture. The Volunteering Network is the final product, which simplified the connection of volunteers and companies, while providing services to facilitate the processes, which are post application system, messaging, task management, engagement tracking, issue reports and a recommendation system.

# TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>v</b>
<b>TABLE OF CONTENTS.....</b>	<b>vi</b>
<b>LIST OF FIGURES .....</b>	<b>viii</b>
<b>LIST OF TABLES .....</b>	<b>ix</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>x</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Problem statement: .....	1
1.2 Objectives: .....	2
1.3 Thesis structure.....	3
<b>CHAPTER 2: RELATED WORKS .....</b>	<b>4</b>
<b>CHAPTER 3: TECHNOLOGIES .....</b>	<b>6</b>
<b>CHAPTER 4: SOFTWARE REQUIREMENTS AND USE CASES.....</b>	<b>8</b>
4.1 SYSTEM FEATURES: .....	8
4.2 USE CASES: .....	18
4.2.1 Volunteer and Company Representative user use-case scenarios .....	20
4.2.2 Volunteer user only use-case scenarios .....	25
4.2.3 Company Representative user only use-case scenarios .....	26
4.2.4 Admin user use-case scenarios .....	29
4.3 NON-FUNCTIONAL REQUIREMENTS: .....	30

<b>CHAPTER 5: SYSTEM DESIGN .....</b>	<b>32</b>
5.1 ARCHITECTURE.....	33
5.2 SYSTEM DECOMPOSITION.....	36
5.2.1 Authentication.....	37
5.2.2 Post management .....	38
5.2.3 Task management .....	39
5.2.4 Messaging .....	40
5.2.5 Database.....	41
5.3 USER INTERFACE.....	42
5.3.1 Sign-up page .....	44
5.3.2 Dashboard .....	45
5.3.3 Post creation.....	46
5.3.4 Task management .....	47
<b>CHAPTER 6: CONCLUSION.....</b>	<b>48</b>
<b>REFERENCES.....</b>	<b>49</b>

## LIST OF FIGURES

Figure 4.1: Feature tree .....	8
Figure 4.2: Use case diagram .....	19
Figure 5.1: Class diagram .....	33
Figure 5.2: Layered architecture example.....	35
Figure 5.3: Serverless architecture example .....	35
Figure 5.4: Sequence diagram for sign-up and login .....	37
Figure 5.5: Sequence diagram for creating a post.....	38
Figure 5.6: Sequence diagram for creating a task .....	39
Figure 5.7: Sequence diagram for sending a message .....	41
Figure 5.8: Database schema .....	42
Figure 5.9: Diagram of all pages in application.....	43
Figure 5.10: Volunteer user sign-up page .....	44
Figure 5.11: Company Representative user sign-up page.....	44
Figure 5.12: Company Representative user dashboard.....	45
Figure 5.13: Volunteer user dashboard .....	45
Figure 5.14: Post creation page .....	46
Figure 5.15: Company Representative user task view page .....	47
Figure 5.16: Volunteer user task view page .....	47



**LIST OF TABLES**

Table 4.1: Non-functional requirements .....30

## **ACKNOWLEDGEMENTS**

Firstly, we would like to thank our respective families and friends who have motivated and backed us throughout our studies. We would also like to express our deepest gratitude for the continuous support of our two mentors, Assoc. Prof. Dr. Kanita Karađuzović-Hadžiabdić, from the International University of Sarajevo, and Alen Seferović, from the company Ant Colony, for helping us achieve our goal with this project with their frequent and valuable inputs.

Nedim Kunovac, Mirza Redžepović, Edin Žiga

June 2023, Sarajevo, Bosnia and Herzegovina

# **CHAPTER 1: INTRODUCTION**

Volunteers are considered to be the pillars of modern society, as their involvement in many projects is considered to be essential for a successful outcome. Whichever reason they choose to engage in a work opportunity, be it a familiar environment, passion project or to simply acquire work experience, the result of their participation will be positive. Additionally, volunteering is a great way for people to have a taste of what it would be like in an actual work environment, where they would constantly communicate with their colleagues and work together towards the same goal. For companies, volunteers represent assets which do not consume a large chunk of company resources, while at the same time providing various skills that could benefit the company immensely. With a good group of volunteers, companies are able to increase their ambitions for a project in terms of scale and outcome. The assistance of the volunteers is in most cases crucial to the final product the company eventually achieves.

## **1.1 Problem statement:**

The need for skilled volunteers and their availability in today's society is more of an issue than one would assume. Companies struggle when the need for volunteers arises, as the only method of recruiting is old-fashioned, time-consuming and expensive. In some cases, rather than simple manpower, companies require knowledge and expertise for a project that is planned. Having the requirements unfulfilled can not only slow down the project at hand, but potentially even prevent it from being completed at all. Additionally, if the project does not fall into the interests of the volunteers, their engagement will be minimal and possibly even futile. All of these issues represent the gap between the

volunteers finding suitable work and the companies recruiting the volunteers that would make the largest impact. This thesis describes the solution of this problem, which will serve as a “bridge” for this gap, where the volunteers and companies would be able to connect and benefit each other to work towards more beneficial outcomes for both of them.

## **1.2 Objectives:**

By taking a process and making it quicker and easier, other aspects benefit greatly as a result. With the creation of a platform which would serve as a tool for both volunteers and companies, they can shift their focus from applications and discussions to getting other, more productive work done. The more companies and volunteers utilize the platform, the more useful it will be. Coming in the form of a cross-platform mobile application, it will be accessible to pretty much everyone who has a mobile phone. Using a very simple, customized interface, it provides everyone with the opportunity to use the platform effectively. The main objectives of this project can be summarized by the following points:

1. To develop a user-friendly platform for matching volunteers with opportunities presented by companies, based on their skills and interests.
2. To create a large database of organizations and their needs, including information on their missions, goals, and the specific skills and expertise they require.
3. Reduce the complexity and duration of the organization aspect of a project or engagement.
4. Explain in detail the architecture and implementation of the platform.

### **1.3 Thesis structure**

This paper consists of the following six chapters: Introduction, Related works, Technologies, Software requirements and use cases, System design and Conclusion. The first chapter, Introduction, gives an overview of the paper and the application at hand, by describing the problem statement, objectives of the thesis and the structure of the paper. Related works chapter explores existing systems which correlate to the topic and how they differ to the one presented in this paper, mostly by pointing out the aspects they lack. The Technologies chapter gives a rundown of the technologies used to develop this platform. Moving on, the next chapter, Software requirements and use cases, lists and describes all of the system features, user requirements, functional requirements, non-functional requirements and use cases of the platform. All features, along with their requirements, and use cases are illustrated using a feature tree and use case diagram, respectively, for better understanding. The System design chapter gives an architectural overview of the platform, as well as a description of the engineered implementation. Using different illustrations, such as sequence diagrams, a class diagram and a database model, as well as including the look and explanation of the user interface, details regarding the implementation can be understood more clearly. Finally, the last chapter, Conclusion, provides a summary of the paper and application, while also pointing out the focus and goal of this thesis.

## CHAPTER 2: RELATED WORKS

This chapter presents the existing platforms, and other media whose functionality or topics of discussion correlate with the topic at hand. Numerous platforms exist that satisfy the needs of connecting volunteers and companies, but they fail to provide functionalities that other platforms do when it comes to project management and tracking progress.

VolunteerMatch (VolunteerMatch, 2023) is a web based online platform that focuses on facilitating connections between volunteers and non-profit organizations. It consists of a comprehensive database of organizations and volunteers, that allows individuals to apply for different jobs, based on their skills, interests and location. While it is a useful tool, it is web based, and has no further functionality besides applying for jobs. Furthermore, idealist.org (idealist, 2023) follows the same concept. It is an online platform that serves to connect individuals, organizations, and companies with volunteering opportunities, jobs, and internships. It's conceptually created as a hub for those aiming to make a positive impact, while streamlining the process of finding and applying for jobs and opportunities. Continuing, LinkedIn (LinkedIn, 2023) is a professional Java, JavaScript and Scala based networking platform primarily aimed at those seeking jobs and carrier opportunities. It allows users to create and manage their profiles for the purpose of job searching, carrier development and business networking. While primarily aimed at professionals, it can also be used for promoting volunteering opportunities, though such occurrences are rare. Finally, Upwork (Upwork, 2023) is a popular online freelancing platform created with AngularJS that connects individuals with various projects, from either companies or other paying individuals. The concept of the

platform allows for any type of job listing to be posted and provides various project management tools and feedback mechanisms to promote effective collaborations. It must be noted that similar platforms exist, such as Freelancer.com (Freelancer, 2023) and Fiverr (Fiverr, 2023), who fall into this category, but provide their own unique features within that space.

While platforms do exist that solve the problem at hand, they don't incorporate the key features needed to make a "all in one" solution. By utilizing the key functionalities of the platforms and solutions presented, it is possible to create a platform that will satisfy all user needs. Beyond the social and business-related benefits that such a platform will provide, it will also promote personal growth, as such opportunities often do.

## CHAPTER 3: TECHNOLOGIES

The platform will be developed using the Flutter framework, which is an open-source and cross-platform framework based on the Dart programming language that enables developers to build mobile, web and desktop applications (Flutter, 2023). Besides its functionality, it provides various customizable widgets, which give developers the ability to create a wide range of proper user interfaces. With these widgets, developers can build components for their applications that can be constantly reused, regardless of the platform. Given its ability to seamlessly transition between Web, Android and Mobile, it was only natural to prefer Flutter for this type of project.

With their seamless integration, the choice of the database fell to Firebase. Firebase is classified as Backend as a Service (BaaS), using which applications can be swiftly and seamlessly deployed (Firebase, 2023). Since both Flutter and Firebase are developed and maintained by Google, they provide a good experience during development, efficiency-wise. Firestore is one part of Firebase, and it operates as a NoSQL document database that uses a document-oriented data model, where the data is stored in documents and can be organized into collections. It contains a set of key-value pairs, where the values can be both simple data types, strings, numbers or booleans, and complex data types, arrays or nested objects (Firestore, 2023).

Furthermore, Firebase provides a variety of services other that can be used in Flutter apps, such as Authentication, Realtime Storage, usage analytics etc. (Firebase, 2023). Firebase Authentication allows for secure registration of users, as well as provides services such as email verification, password resetting, managing access privileges and linking with other



Google's services (Firebase Authentication, 2023). Continuing, Realtime Storage provides developers with the ability to store files on Google's cloud servers, which can then be referenced so that they may be used later (Firebase Realtime Database, 2023). Lastly, with usage analytics, it is possible to view detailed data of the oncoming and outgoing traffic, as well as which part of the platform is used the most (Firebase, 2023).

## CHAPTER 4: SOFTWARE REQUIREMENTS AND USE CASES

This chapter gives a detailed overview of the requirements of this application, along with the actors and use cases who are listed and described in detail and visualized with a use case diagram. The requirements are usually a representation of what the client wants and clearly stating what is expected, while for developers they serve as a roadmap of what they will implement. Their purpose in this thesis is to give an overview of the features and functionalities within the platform. Figure 4.1 represents a good visualization of the features and requirements in the form of a feature tree.

### 4.1 SYSTEM FEATURES:

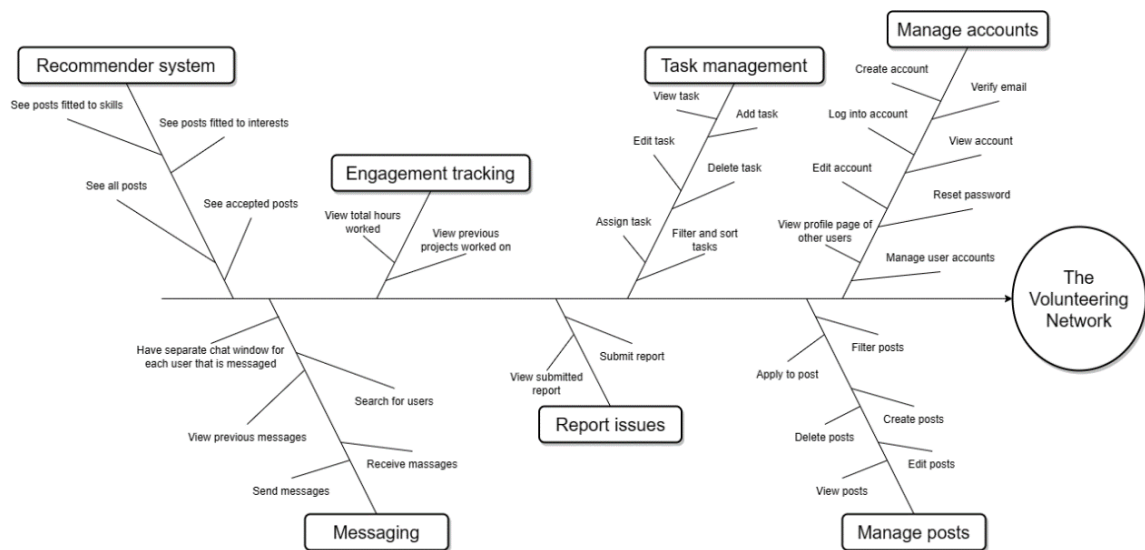


Figure 4.1: Feature tree

FTR1: Manage accounts: The user is able to create, view, edit and use their created account. (UR1.1-UR1.6 are specific to both Volunteer and Company Representative users; UR1.7 is specific to Company Representative user; UR1.8 is specific to Admin user)

UR1.1: Create account (Must have)

FR1.1.1: Users are able to add their full name, email, password, date of birth and a profile photo to their volunteer account.

FR1.1.2: Users are able to add skills they possess to their volunteer account.

FR1.1.3: Users are able to add the topics which are of interest to their volunteer account.

FR1.1.4: Users are able to add their company title, name of representative, photo containing the logo of the company, VAT number, email and password to their company representative account.

FR1.1.5: Users are redirected to the login page where their credentials can be entered.

FR1.1.6: Users are not able to create an account with an email that is already in use.

UR1.2: Verify email (Should have)

FR1.2.1: Users receive an email which contains the verification link for the newly created account.

FR1.2.2: Users are able to verify their email by clicking the link within the content of the email.

FR1.2.3: Users are redirected to their profile page where a notification is displayed confirming that the email was verified.

UR1.3: Log into account (Must have)

FR1.3.1: Users are able to access their account by using their email and password.

FR1.3.2: Users are not able to access their account if the email has not been verified.

UR1.4: View account (Must have)

FR1.4.1: Users are able to see the general information which they entered upon sign-up.

FR1.4.2: Users are able to see their profile photo they have chosen.

FR1.4.3: Users are able to see the skills and interests they have chosen.

UR1.5: Edit account (Must have)

FR1.5.1: Users are able to view the information connected to their account when entering the profile settings page.

FR1.5.2: Users are able to change their profile photo by uploading another image.

FR1.5.3: Users are able to change their name, date of birth and general info by editing text boxes containing each piece of information in the profile settings page, respectively.

FR1.5.4: Users are able to add and remove skills as tags in the profile settings page.

FR1.5.5: Users are able to add and remove interests as a checklist in the profile settings page.

UR1.6: Reset password (Could have)

FR1.6.1: Users are able to click on the “Forgot password” link on the login page to initiate the process of resetting the password.

FR1.6.2: Users are redirected to a page where they provide their email.

FR1.6.3: Users are able to submit their email in order to receive the reset password link.

FR1.6.4: Users are able to enter a new password and confirm password after being redirected by clicking the reset password link sent to them.

FR1.6.5: User is redirected to login page where he/she can enter the updated credentials.

UR1.7: View profile pages of other users (Could have)

FR1.7.1: Users are able to view profiles of other users that have registered.

FR1.7.2: Users are able to view basic information of other users on their respective profile page.

UR1.8: Manage user accounts (Could have)

FR1.8.1: Users are able to add and delete users using the admin panel.

FTR2: Manage posts: The user is able to view, edit, delete, create, update and apply to posts. (UR2.1 and UR2.6 are specific to both Volunteer and Company Representative users; UR2.2-UR2.4 are specific to Company Representative user; UR2.5 is specific to Volunteer user)

UR2.1: View posts (Must have)

FR2.1.1: Users are able to view posts for opportunities on the dashboard.

FR2.1.2: Users are able to have a detailed view of the information in the post.

FR2.1.3: Users are able to view the company that created the post.

UR2.2: Edit posts (Must have)

FR2.2.1: Users are able to edit their own posts.

FR2.2.2: Users are able to modify the entered information for the post, such as title, description, start date, end date, due date, number of accepted applicants and opportunities.

FR2.2.3: Users are able to modify the skills required.

FR2.2.4: Users are able to change the category the post fits in.

UR2.3: Delete posts (Must have)

FR2.3.1: Users are able to delete their created post.

FR2.3.2: All tasks related to the post are also deleted.

UR2.4: Create posts (Must have)

FR2.4.1: Users are able to add a post that will appear on the dashboard.

FR2.4.2: Users are able to add information such as title, description, start date, end date, due date, number of accepted applicants and opportunities that are provided.

FR2.4.3: Users are able to add the required skills related to that post.

FR2.4.4: Users are able to add the category the post fits in.

UR2.5: Apply to post (Must have)

FR2.6.1: Users are able to apply to the post they are interested in.

FR2.6.2: Users are able to view the pending applications.

UR2.6: Filter and sort posts (Could have)

FR2.6.1: Users are able to sort posts by date created and due date.

FR2.6.2: Users are able to filter posts by company and date.

FTR3: Task management: The user is able to view, add, edit and delete tasks that pertain to an engagement. (UR3.1 and UR3.6 are specific to both Volunteer and Company Representative users; UR3.2-UR3.5 are specific to Company Representative user)

UR3.1: View task (Should have)

FR3.1.1: Users are able to view tasks relevant to them.

FR3.1.2: Users are able to see the task title, description, due date and assignee.

FR3.1.3: Users are able to click on a task for a detailed view of it.

UR3.2: Add task (Should have)

FR3.2.1: Users are able to add a task to an existing post.

FR3.2.2: Users are able to add information to task, such as title, due date and description.

UR3.3: Edit task (Should have)

FR3.3.1: Users are able to view the information already entered for the task.

FR3.3.2: Users are able to change the task title, due date and assignee.

FR3.3.3: Users are able to view the new information entered after saving the changes.

UR3.4: Delete task (Should have)

FR3.4.1: Users are able to delete a specific task from the task list related to a post.

UR3.5: Assign task (Should have)

FR3.5.1: Users are able to assign a task to a specific user.



UR3.6: Filter and sort tasks (Could have)

FR3.6.1: Users are able to sort the tasks by name and due date.

FR3.6.2: Users are able to filter the tasks by date.

FTR4: Report issues: The user is able to report problems and submit reports to be reviewed. (UR4.2 is specific to both Volunteer and Company Representative users; UR4.1 is specific to Admin user)

UR4.1: View submitted reports (Could have)

FR4.1.1: Users are able to see submitted user reports regarding issues.

FR4.1.2: Users are able to classify reports as solved or unsolvable.

UR4.2: Submitting the report (Should have)

FR4.2.1: Users are able to submit reports that refer to an issue.

FR4.2.2: Users are able to add a title and a brief description to the issue report.

FTR5: Tracking engagement: The user is able to track the activity and engagement of other users and his/her own. (UR5.1 and UR5.2 are specific to both Volunteer and Company Representative users)

UR5.1: View total hours worked (Could have)

FR5.1.1: Users are able to view total hours accumulated by other users on their respective profile.

FR5.1.2: Users are able to view their own total hours accumulated on their profile page.

UR5.2: View previous projects worked on (Could have)

FR5.2.1: Users are able to view previous projects worked on by other users on their respective profile.

FR5.2.2: Users are able to view their own previous projects worked on on their profile page.

FTR6: Messaging: The user is able to communicate via a messaging component with other users. (UR6.1-UR6.5 are specific to both Volunteer and Company Representative users)

UR6.1: Send messages (Should have)

FR6.1.1: Users are able to send messages to other users on the messaging page.

FR6.1.2: Users are able to view the message they have sent, in the chat window.

UR6.2: Receive messages (Should have)

FR6.2.1: Users are able to receive messages from other users, which will show up on the messaging page.

FR6.2.2: Users are able to view the message they have received, in the chat window.

UR6.3: View previous messages (Could have)

FR6.3.1: Users are able to view previously sent and received messages on the messaging page.

FR6.3.2: Users are able to distinguish previously sent and received messages.

UR6.4: Search for users (Could have)

FR6.4.1: Users are able to search for any other registered users on the messaging page in order to message them.

UR6.5: Have separate chat window for each user that is messaged (Could have)

FR6.5.1: Users are able to have separate windows for each user they have sent and/or received a message to/from.

FTR7: Recommender system: The user is able to view posts that are fitted to his/her skills and interests. (UR7.1-UR7.4 are specific to Volunteer user)

UR7.1: See posts fitted to skills (Should have)

FR7.1.1: Users are able to filter posts based on the skills they have attributed to their profile.

UR7.2: See posts fitted to interests (Should have)

FR7.2.1: Users are able to filter posts based on the interests they have attributed to their profile.

UR7.3: See all posts (Must have)

FR7.3.1: Users are able to view all posts that are active, without regards to skills or interests.

UR7.4: See accepted posts (Must have)

FR7.4.1: Users are able to view all posts to which they have been accepted.

## **4.2 USE CASES:**

This section presents flows of use cases that are derived from the user requirements. There are four groups of use cases: Volunteer and Company Representative user use-cases, Volunteer user only use-cases, Company Representative user only use-cases and Admin user use-cases. To better understand the relationships between the user types and use cases, a use case diagram is provided in Figure 4.2.

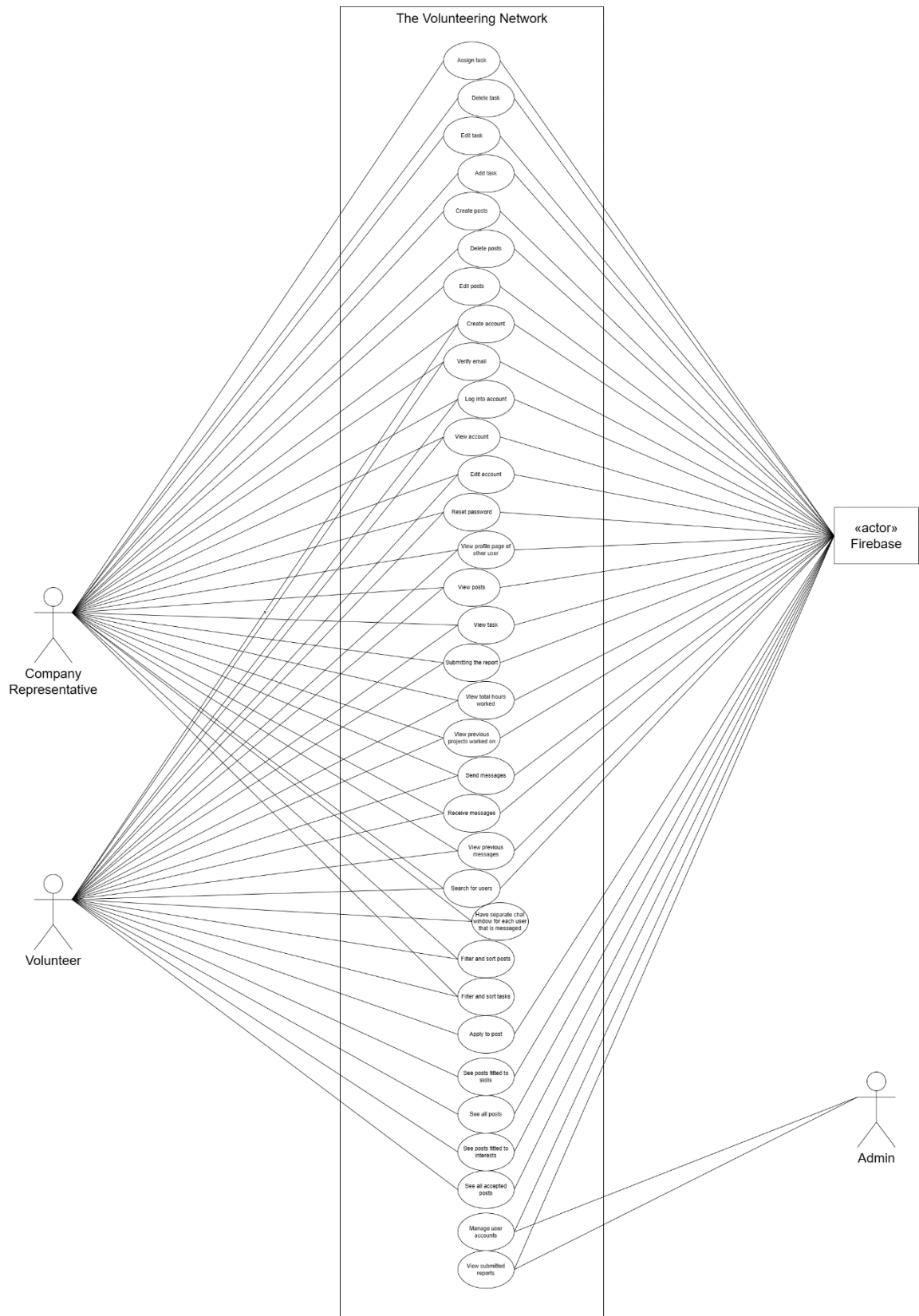


Figure 4.2: Use case diagram

#### 4.2.1 Volunteer and Company Representative user use-case scenarios

##### *UC-1.1: Create account*

**Description:** User can create an account as a company or volunteer.

**Priority:** High

**Preconditions:** Application is running; User is connected to the internet

**Postconditions:** An account is created with the information provided by the user.

**Basic Flow:**

1. User clicks on the 'Sign up' button.
2. System displays the page with user types.
3. User clicks on image of company or volunteer, depending on the type of user he/she wishes to create.
4. System displays a page containing a form that requires relevant information for account creation.
5. User enters the information required and submits the form.
6. System redirects the user to the login page.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. The information will not be accepted and the system will display an appropriate error message.

##### *UC-1.2: Verify email*

**Description:** User can edit their account which allows them to change the account information.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** Email of the user is verified.

**Basic Flow:**

1. User receives an email containing a verification link after creating an account.
2. User click the verification link in the said email.
3. User is redirected to a page where a message saying "Your email has been verified" appears

**What can go wrong:** User is able to click the link while already having verified his email. The page will display a message saying "Your email has already been verified."

##### *UC-1.3: Log into account*

**Description:** User can login to their created account using the email and password entered on signing up.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is not logged in; User has created an account

**Postconditions:** User accessed created account.

**Basic Flow:**

1. User clicks on login button on the page upon opening the application.
2. System redirects to login page.
3. User enters email and password.
4. User clicks login button on login page.

5. System checks the entered information.
6. System redirects user to the dashboard page.

**What can go wrong:** User did not enter the needed information properly. The system displays a popup with an appropriate error message.

#### *UC-1.4: View account*

**Description:** User can view their account.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** User's account information is displayed.

**Basic Flow:**

1. User clicks on the profile page button on the bottom navigation tab.
2. System displays profile page.

#### *UC-1.5: Edit account*

**Description:** User can edit their account which allows them to change the account information.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** The user's account information is changed.

**Basic Flow:**

1. User clicks on the profile settings button located in other options.
2. System displays the profile settings page.
3. User edits the fields that were subject to be changed.
4. User clicks save changes button.
5. System saves new profile information.
6. System displays a popup containing appropriate message to show changes were made.
7. System redirects user to profile page.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. Changes to account information will not be made and the system will display an appropriate error message.

#### *UC-1.6: Reset password*

**Description:** User can reset the password that was used to create an account.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet;

**Postconditions:** The password of the user is reset.

**Basic Flow:**

1. User clicks on the reset password button on the page upon opening the application.
2. System displays reset password page.
3. User enters the email used when creating the account.
4. System sends an email containing a reset password link.
5. User clicks on the link in the email received.
6. System redirects user to reset password page where new password is required.
7. User enters new password.

8. System changes the password of that user.

**What can go wrong:** The user may not enter the information required properly. The system displays an appropriate error message.

*UC-2.1: View posts*

**Description:** User can see previously created posts.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; At least one post has been created

**Postconditions:** User has seen posts that were previously created.

**Basic Flow:**

1. User chooses a post to view from the dashboard.
2. User clicks on the chosen post.
3. System displays page with detailed information about the post.

*UC-2.6: Filter posts*

**Description:** User can filter posts.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in; At least one post has been created

**Postconditions:** User views posts based on the filter chosen.

**Basic Flow:**

1. User chooses a filter on the dashboard.
2. User clicks on the chosen filter.
3. System displays page with filtered posts based on the choice of the user.

*UC-3.1: View task*

**Description:** User can see previously created task.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; At least one post has been created; At least one task has been created

**Postconditions:** User has seen task that was previously created.

**Basic Flow:**

1. User finds post for which there have been accepted users.
2. User chooses a post to view from the choices.
3. User clicks on the chosen post.
4. System displays page with detailed information about the post.
5. User clicks on the chosen task.
6. System displays page with detailed information about the task.

*UC-3.6: Filter and sort tasks*

**Description:** User can filter and sort tasks.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in; At least one post has been created; At least one task has been created



**Postconditions:** User views tasks based on the filter and sorting chosen.

**Basic Flow:**

1. User finds post for which there have been accepted users.
2. User chooses a post to view from the choices.
3. User clicks on the chosen post.
4. System displays page with detailed information about the post.
5. User chooses a filter on the page.
6. User clicks on the chosen filter.
7. User chooses a sorting type on the page.
8. User clicks on the chosen sorting type.
9. System displays page with filtered and sorted tasks based on the choice of the user

*UC-4.2: Submitting the report*

**Description:** User can submit an issue report.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** User has submitted an issue report.

**Basic Flow:**

1. User clicks on profile page.
2. System displays profile page.
3. User clicks on other options button.
4. System displays dropdown with options.
5. User clicks on “Report issue” option.
6. System displays popup window containing relevant input fields.
7. User enters relevant information inside of the fields.
8. User submits the report by clicking the button to confirm the submission.
9. System displays popup with the appropriate message.

*UC-5.1: View total hours worked*

**Description:** User can view total hours worked.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** User has viewed the total hours worked on previous posts.

**Basic Flow:**

1. User clicks on profile page.
2. System displays profile page.
3. User searches for relevant field on profile page.

**Alternate Flow (for Company Representative):**

1. User clicks on a post on the dashboard.
2. System displays page with detailed information about the post.
3. User clicks on button “Applicants”.
4. System displays page containing applicants for that post.
5. User chooses applicant whose profile is to be viewed.
6. User clicks on chosen applicant.
7. System displays profile page of the chosen applicant.
8. User searches for relevant field on profile page.

*UC-5.2: View previous projects worked on*

**Description:** User can view previous projects worked on.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** User has viewed the previous projects worked on.

**Basic Flow:**

1. User clicks on profile page.
2. System displays profile page.
3. User searches for relevant field on profile page.

**Alternate Flow (for Company Representative):**

1. User clicks on a post on the dashboard.
2. System displays page with detailed information about the post.
3. User clicks on button “Applicants”.
4. System displays page containing applicants for that post.
5. User chooses applicant whose profile is to be viewed.
6. User clicks on chosen applicant.
7. System displays profile page of the chosen applicant.
8. User searches for relevant field on profile page.

*UC-6.1: Send messages*

**Description:** User can send messages to other users.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has messaged at least one user

**Postconditions:** User has sent a message to another user.

**Basic Flow:**

1. User clicks on messaging page.
2. System displays messaging page.
3. User clicks on user to send a message to.
4. System displays chat window for the chosen user.
5. User enters message.
6. User clicks send icon.
7. System displays sent message inside chat window.

*UC-6.2: Receive messages*

**Description:** User can receive messages from other users.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has received message from at least one user

**Postconditions:** User has seen received message from another user.

**Basic Flow:**

1. User clicks on messaging page.
2. System displays messaging page.
3. User clicks on user from which a message has been received.
4. System displays chat window for the chosen user.
5. User sees received message.

Use cases *UC-6.3: View previous messages* and *UC-6.5: Have separate chat window for each user that is messaged* are like use case *UC-6.2: Receive messages* since these functionalities are displayed on the same page and have a similar purpose.

*UC-6.4: Search for users*

**Description:** User can search for user to message.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in

**Postconditions:** User has searched for another user.

**Basic Flow:**

1. User clicks on messaging page.
2. System displays messaging page.
3. User clicks on search button.
4. User enters the name of the user to search for.
5. User confirms the choice.
6. System displays chat window for the chosen user.

#### 4.2.2 Volunteer user only use-case scenarios

*UC-2.5: Apply to post*

**Description:** User can apply to a post.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; At least one post has been created

**Postconditions:** User has applied to a chosen post.

**Basic Flow:**

1. User chooses a post to view from the dashboard.
2. User clicks on the chosen post.
3. System displays page with detailed information about the post.
4. User click button to apply to post.
5. System displays popup with the appropriate message.

*UC-7.1: See posts fitted to skills*

**Description:** User can view only posts that are fitted to the previously entered skills.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; There is at least one post created

**Postconditions:** The user views only posts that match their skills.

**Basic Flow:**

1. User clicks the button “By Skills” on the dashboard page.
2. System refreshes page and shows posts only matching the skills of the user.

*UC-7.2: See posts fitted to interests*

**Description:** User can view only posts that are fitted to the previously entered interests.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; There is at least one post created

**Postconditions:** The user views only posts that match their interests.

**Basic Flow:**

1. User clicks the button “By Category” on the dashboard page.
2. System refreshes page and shows posts only matching the interests of the user.

*UC-7.3: See all posts*

**Description:** User can view all available posts.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; There is at least one post created

**Postconditions:** The user views all posts that are available.

**Basic Flow:**

1. User clicks the button “All” on the dashboard page.
2. System refreshes page and shows all posts available to the user.

*UC-7.4: See accepted posts*

**Description:** User can view all posts for which the users’ application was successful.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; There is at least one post created; User is accepted to at least one post

**Postconditions:** The user views all posts for which the users’ application was successful.

**Basic Flow:**

1. User clicks the button “Accepted” on the dashboard page.
2. System refreshes page and shows all posts for which the users’ application was successful.

#### **4.2.3 Company Representative user only use-case scenarios**

*UC-1.7: View profile pages of other users*

**Description:** User can view profile pages of other users that created an account.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has created a post; User has applicants on post

**Postconditions:** The user views the profile page of another user

**Basic Flow:**

1. User opens the created post.
2. System displays the relevant information about the post.
3. User clicks applicants’ button.
4. System displays the applicants for that specific post.
5. User clicks on one of the users who have applied.
6. System displays the profile page of that user.

#### *UC-2.2: Edit posts*

**Description:** User can edit their post which allows them to change the relevant information.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has a created post

**Postconditions:** The user's post information is changed.

**Basic Flow:**

1. User clicks on a post created by them on the dashboard.
2. System displays page with detailed information about the post.
3. User clicks on edit post button.
4. System displays popup asking to confirm.
5. User confirms that the post is to be edited.
6. System displays form with input fields for all information required by a post.
7. User changes the information that was due to be changed.
8. User clicks button to save changes made.
9. System redirects user to dashboard.
10. System displays the new information for the post on the dashboard.
11. System displays popup with the appropriate message.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. Changes to post information will not be made and the system will display an appropriate error message.

#### *UC-2.3: Delete posts*

**Description:** User can delete their post.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has a created post

**Postconditions:** The user's post is deleted.

**Basic Flow:**

1. User clicks on a post created by them on the dashboard.
2. System displays page with detailed information about the post.
3. User clicks on delete post button.
4. System displays popup asking to confirm.
5. User confirms that the post is to be deleted.
6. System removes the post from the dashboard.
7. System displays popup with the appropriate message.

#### *UC-2.4: Create posts*

**Description:** User can create a post which becomes open for applying.

**Priority:** High

**Preconditions:** Server is online; User is connected to the internet; User is logged in;

**Postconditions:** The user adds a post.

**Basic Flow:**

1. User clicks on "+" button in bottom right corner.
2. System displays popup asking to confirm the intention of creating a post.
3. User confirms the intention.
4. System displays form with input fields for all information required by a post.

5. User enters all of the required information.
6. User clicks button to create the post.
7. System redirects user to dashboard.
8. System adds the post to the dashboard.
9. System displays popup with the appropriate message.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. Post will not be created and the system will display an appropriate error message.

#### *UC-3.2: Add task*

**Description:** User can add a task to a previously created post.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has a created post; User has accepted applicants

**Postconditions:** The user adds a task to post.

**Basic Flow:**

1. User clicks on previously created post.
2. System displays page with relevant information for that post.
3. User clicks box which contains the text "+ Add".
4. System displays form with input fields for all information required to create the task.
5. User enters all of the required information.
6. User clicks button to create the task.
7. System redirects displays page with relevant information for that post.
8. System adds the task to the page.
9. System displays popup with the appropriate message.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. Task will not be created and the system will display an appropriate error message.

Use case *UC-3.5: Assign task* is like *UC-3.2: Add task* since the task cannot be created without assigning a Volunteer user to it.

#### *UC-3.3: Edit task*

**Description:** User can edit a task which was previously created.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has a created post; User has accepted applicants; User has created a task

**Postconditions:** The information of the task is changed.

**Basic Flow:**

1. User clicks on icon representing a pen on a previously created task.
2. System displays popup asking to confirm.
3. User confirms that the task is to be edited.
4. System displays form with input fields for all information required by a task.
5. User changes the information that was due to be changed.
6. User clicks button to save changes made.
7. System redirects displays page where the tasks are listed.
8. System shows new information for the task on the page.

9. System displays popup with the appropriate message.

**What can go wrong:** The user may not enter required input fields or may enter invalid data. Task will not be created and the system will display an appropriate error message.

#### *UC-3.4: Delete task*

**Description:** User can delete a previously created task.

**Priority:** Medium

**Preconditions:** Server is online; User is connected to the internet; User is logged in; User has a created post; User has accepted applicants; User has created a task

**Postconditions:** The task for the post is deleted.

**Basic Flow:**

1. User clicks on icon representing a bin on a previously created task.
2. System displays popup asking to confirm.
3. User confirms that the task is to be deleted.
4. System removes the task from the page.
5. System displays popup with the appropriate message.

### **4.2.4 Admin user use-case scenarios**

#### *UC-1.8: Manage user accounts*

**Description:** User can manage all user accounts.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in into the admin panel

**Postconditions:** The user views, adds or removes someone's account.

**Basic Flow:**

1. User clicks on the option 'Users'.
2. System displays page containing all users.
3. User does the operation he/she wanted on the user/users by clicking on the respective button.
4. User clicks the save button.
5. System saves the changes the user made.
6. System displays a message saying that account information is changed successfully.

#### *UC-4.1: View submitted reports*

**Description:** User can view submitted issue reports from other users.

**Priority:** Low

**Preconditions:** Server is online; User is connected to the internet; User is logged in into the admin panel

**Postconditions:** The user views and/or reviews reports from other users.

**Basic Flow:**

1. User clicks on the option 'Issue Reports'.
2. System displays page containing all submitted reports.
3. User enters the necessary information that they want to change.
4. User clicks the save button.

5. System saves new account information to the database.
6. System displays a message saying that account information is changed successfully.

### 4.3 NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements represent different constraints that are put on the development of the system in order to provide some additional value to the software. To put it simply, they define how a system is supposed to be, while the functional requirements define what a system is supposed to do. The non-functional requirements for The Volunteering Network are provided in Table 4.1.

*Table 4.1: Non-functional requirements*

Requirement	Definition	Details
NFR1: Hardware Interface	Minimum hardware requirements for the hardware from which the user will use the platform.	OS: Windows 10 and above, Linux, MacOS, Android, IOS CPU: Pentium 4 and above Disk space: 2 GB RAM: 4GB
NFR2: Communication Interface	The user needs an internet connection to access both the mobile and web application	Internet connection.
NFR3: Performance	Speed and responsiveness of the platform and the amount of traffic it can sustain.	PER-1: Applying to post should not take longer than 2 seconds for confirmation message.  PER-2: Retrieving data from the database should not take longer than 1 second.  PER-3: Recommended posts should not take longer than 1 second to load.
NFR4: Availability	Availability of the application for various users	AVL-1: System shall be always available.  AVL-2: Excluded from the calculation of availability is down-time that is reserved for maintenance which happens between 1:00 A.M.



		through 2 A.M. Central European Time once a week.
NFR5: Usability	How easy is it to use the application?	USY-1: Users are expected to be able to navigate and use the application after 5 minutes of usage.
NFR6: Security	Security expectations of the system.	SEC-1: Only admin users can manage users and view issues and vulnerabilities.
NFR7: Design and implementation constraints	Constraints on the design and implementation of the application.	CON-1: The application should be supported for IOS and Android.  CON-2: Non-relational database shall be used.
NFR8: Internationalization and localization.	System availability in different languages and accessibility.	IL-1: The application should also be available in the Bosnian language.

## **CHAPTER 5: SYSTEM DESIGN**

This chapter focuses on the architecture and implementation of the system in detail. The architecture is explained and discussed, as well as the reason behind choosing the architecture in question. By decomposing the system into different modules, they can be examined thoroughly from a technical standpoint, as well as completely going over the implementation. Additionally, the database structure and the system flows are illustrated using a database model and a combination of sequence diagrams, respectively. Finally, parts of the user interface are explained in order to better grasp the picture of the platform. To illustrate the implementation as a whole, a class diagram is provided in Figure 5.1.

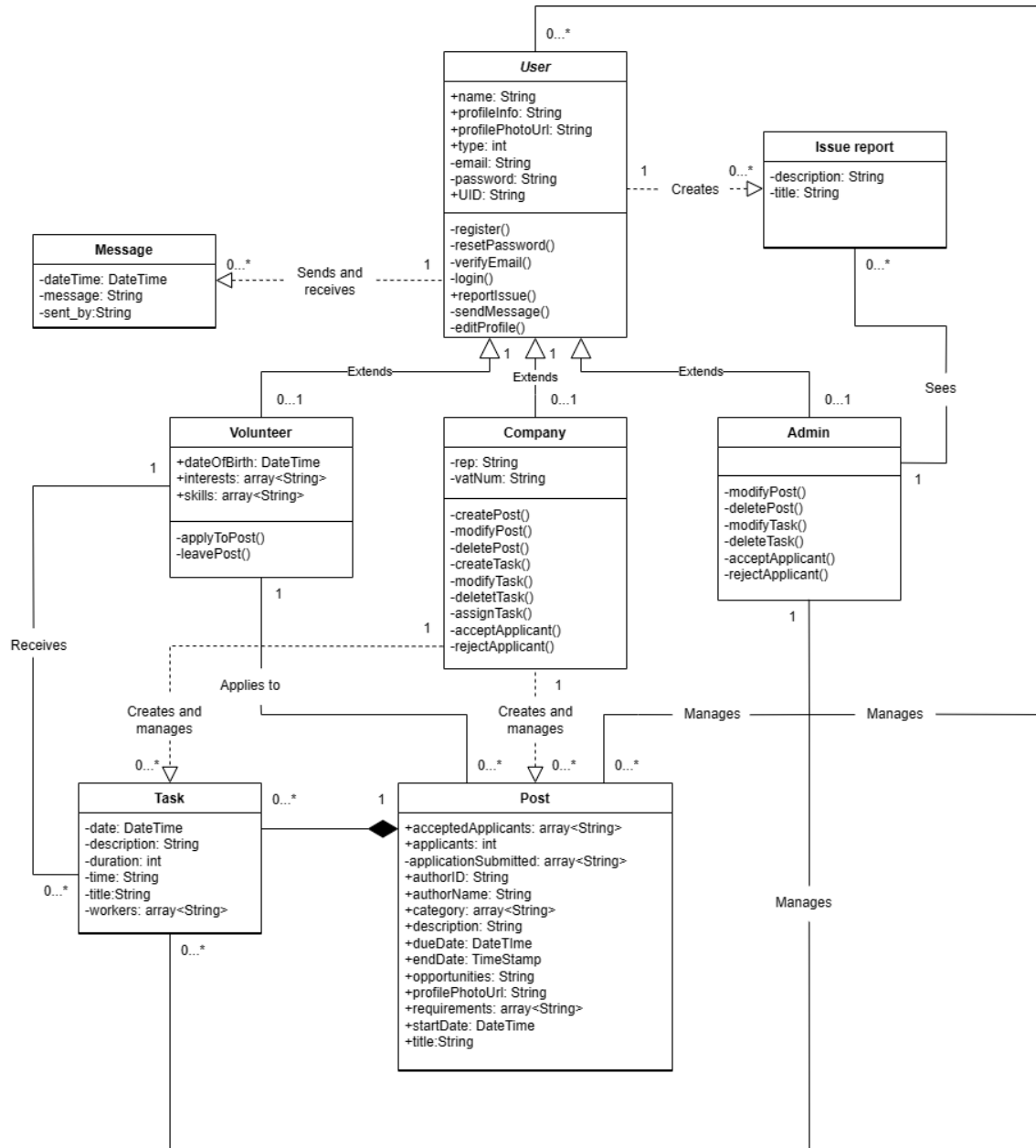


Figure 5.1: Class diagram

## 5.1 ARCHITECTURE

The architecture of The Volunteering Network can be viewed as a hybrid between a serverless and layered architectures. A serverless architecture allows the development to go on without the need of having to manage the underlying structure, while a cloud provider does the work of provisioning servers to run their applications, databases, and

storage systems at any scale. Therefore, the scalability of the serverless architecture is a great benefit when the traffic increases. Additionally, the release cycles are expected to increase, considering more work can be done to the application itself (Datadog, n.d.). On the other hand, a layered architecture represents the application as a pattern of horizontal layers which function together for a single application. Each layer is a different component or block of code and every one attributes something different to the system overall. Since each layer is separated, testing becomes much easier to do. Each layer can be tested independently. Furthermore, dependency is reduced since each layer acts on its own (Baeldung, 2021).

In the case of The Volunteering Network, three layers are implemented: presentation, application and database. The presentation layer represents the user interface as a combination of all UI elements, such as different pages and UI components. Inside of the application layer, we have all of the business logic behind the methods used within the system, such as the creation of posts, tasks, etc. Finally, the database layer represents anything to do with handling the database. In order to utilize the serverless architecture, Firebase was used as the platform of choice. Using the Cloud Functions provided, which are useful when communicating with the database upon certain events within the application, the Firebase cloud servers respond automatically, while also scaling to the needs of the application. This would mean, theoretically, that an infinite number of requests can be executed and Firebase will cover all of the worries regarding scalability. In Figure 5.2 (TechTarget, 2021), we can see an example of a layered architecture.

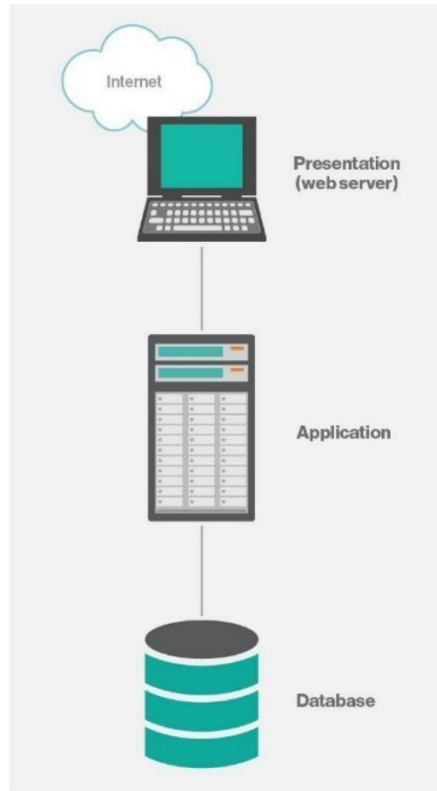


Figure 5.2: Layered architecture example

While in Figure 5.3 (Firebase Tutorials, n.d.) we see a serverless architecture illustrated.

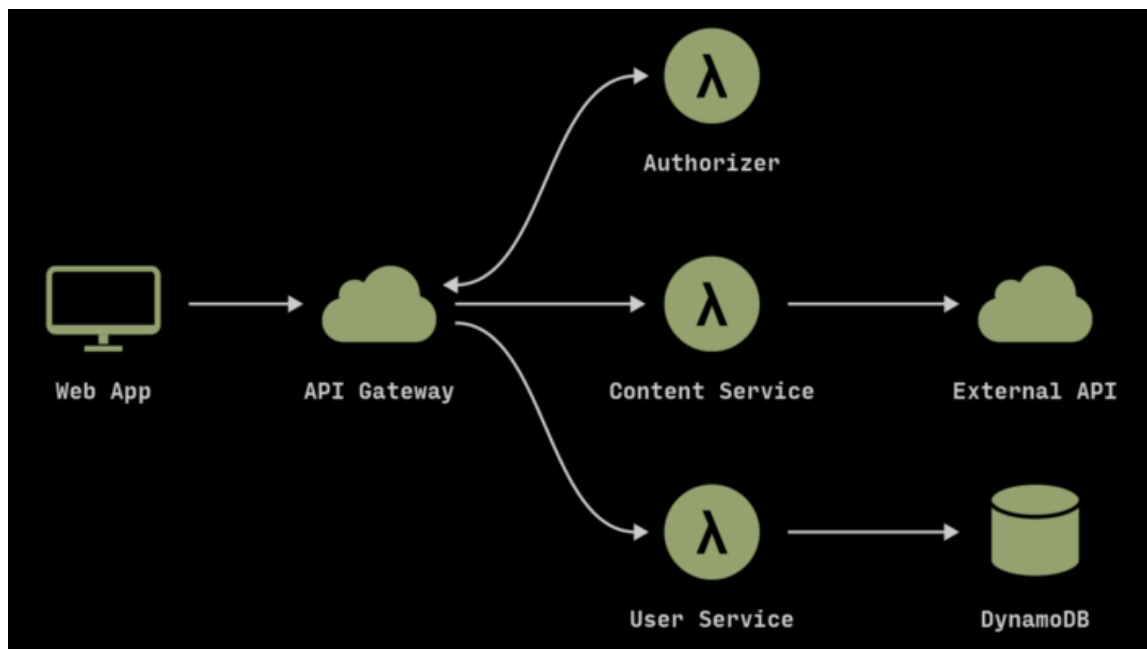


Figure 5.3: Serverless architecture example

## 5.2 SYSTEM DECOMPOSITION

The Volunteering Network is a platform which allows volunteers and companies to apply to volunteering opportunities and provide them, respectively. Depending on the user type, i.e., Volunteer or Company Representative, the displayed content differs slightly, as volunteers can view all posts from companies, while a company can view only its own posts. At its core, the content of the platform consists of posts from various companies that can contain unlimited combinations of the post category and the requirements for the post in the form of skills. Each post can belong to only one category. Each post has its own requirements. One post can have multiple skills as requirements. The Volunteer users are supposed to browse the posts, find one they find fitting and apply to it, while the Company Representative user creates the post and determines whether to accept the application on that specific post. On top of that, each post contains a task management page where tasks can be created and assigned by the Company Representative user to the Volunteer user. Additionally, all users are able to message each other without restrictions based on the user type. The users are managed by the Admin user through the admin panel, in case of unexpected events.

For better understanding, the system can be divided into five different modules: Authentication, Post management, Task management, Messaging and Database. Each module is explored in more detail, in order to have a clearer understanding.

### 5.2.1 Authentication

The Volunteering Network provides users with the ability to authenticate themselves using the email and password used upon creating their account. The service used for authentication was Firebase Authentication, where the user has its own collection added to the Users collection inside the database. When the user enters the email and password, the server compares them to the email and hashed password stored in the database. If the authentication is successful, the user is granted access to the application. In Figure 5.4, the sign-up and login are illustrated using a sequence diagram.

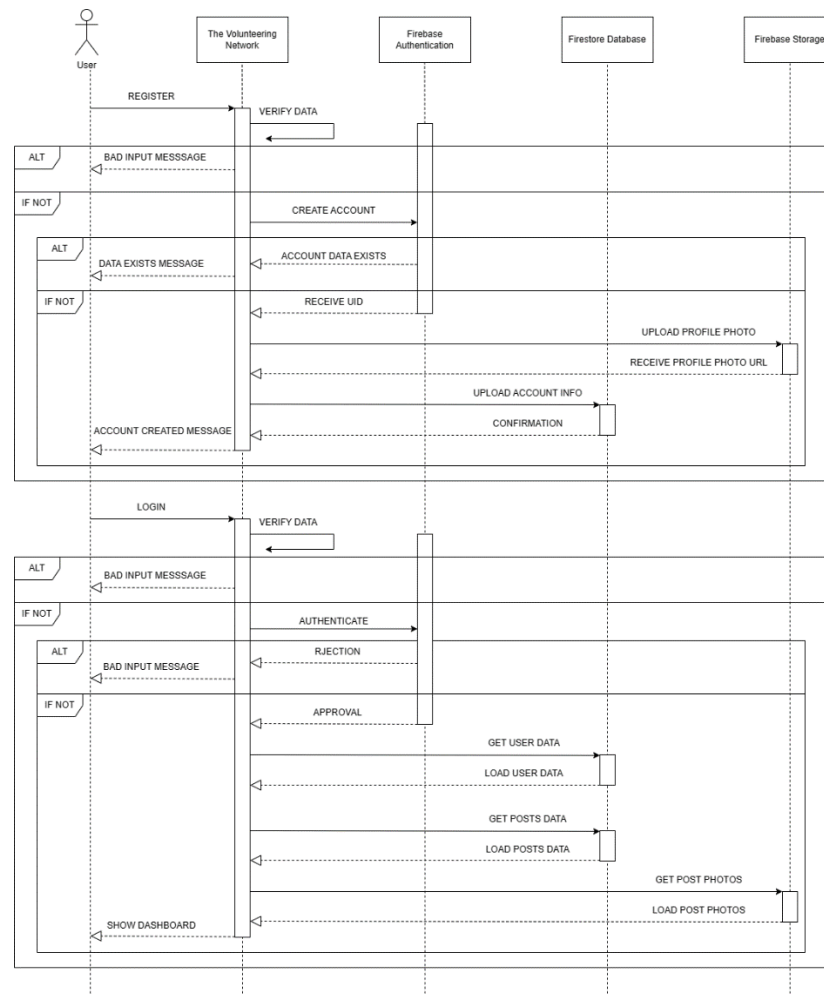


Figure 5.4: Sequence diagram for sign-up and login

### 5.2.2 Post management

The post management module of the application consists of all functionalities related to posts. Focusing on the posts themselves, manipulating them is exclusive to Company Representative users. They can create posts which will become available for Volunteer users to apply to. Additionally, Company Representative users are able to edit and delete their previously created posts. All changes will be visible to both the Volunteer and Company Representative users. Every post is within a collection in the database named “Posts” and contains all relevant information. Each post represents a document within the collection and is pulled whenever some information is needed from it. In Figure 5.5, the process of creating a post is illustrated using a sequence diagram.

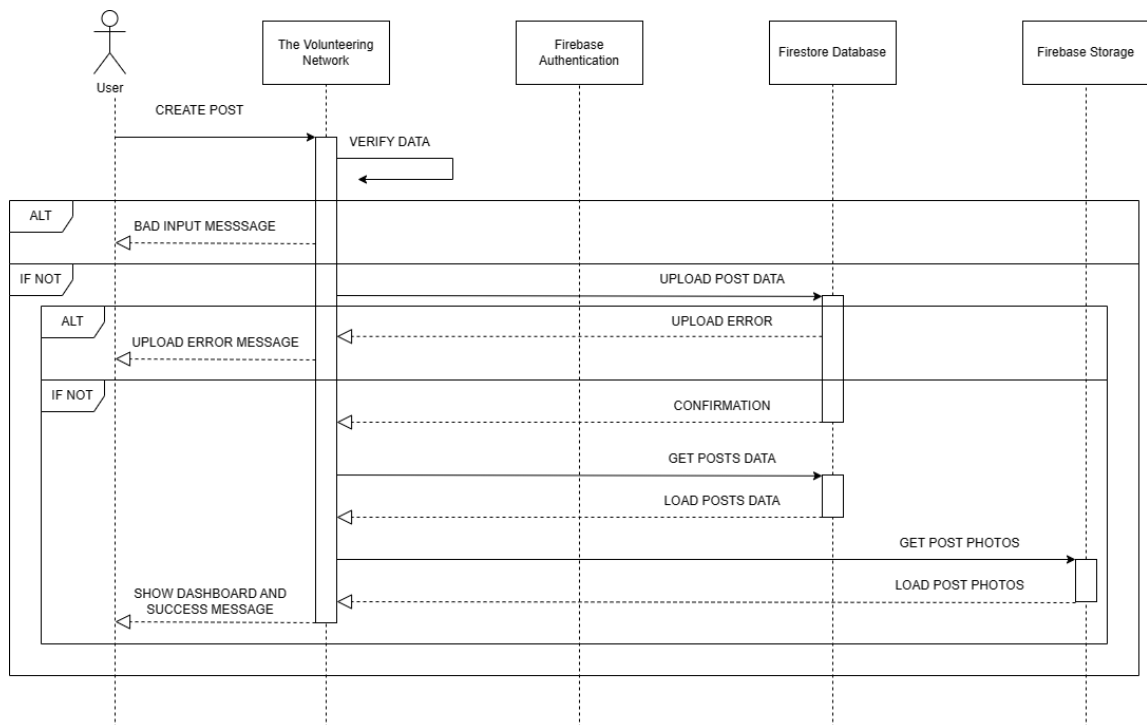


Figure 5.5: Sequence diagram for creating a post



### 5.2.3 Task management

The task management module of the application consists of all functionalities related to tasks. Focusing on the tasks themselves, manipulating them is exclusive to Company Representative users. They can create and assign tasks for chosen Volunteer users that have been accepted to that post. Additionally, Company Representative users are able to edit and delete their previously created tasks. Any changes to the task will be visible to the Company Representative user and the Volunteer user to which the task is assigned to. Tasks in the database are a subcollection of each post document. Each task in that collection is its own document containing all of the relevant information for it. As is the case with posts, the entire document is pulled when some information needs to be retrieved. In Figure 5.6, the process of creating a task is illustrated using a sequence diagram.

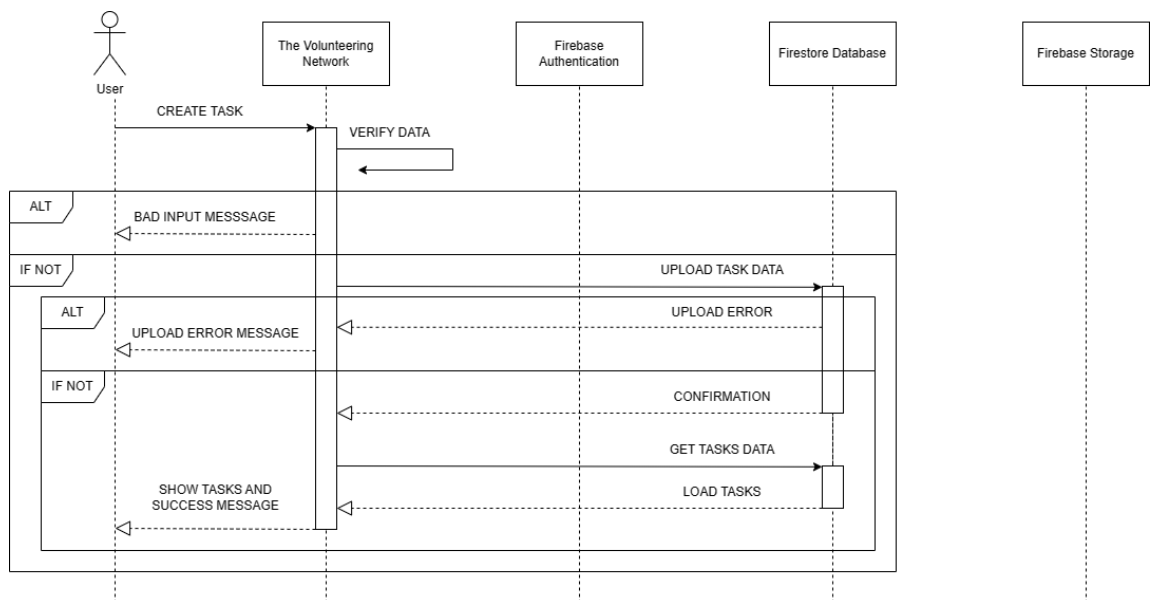


Figure 5.6: Sequence diagram for creating a task

#### 5.2.4 Messaging

The Real-time Messaging aspect of The Volunteering Network was implemented by utilizing Flutter's built-in Stream functions and nesting unique collections within Firestore. One of the key components of Flutter's framework is the StreamBuilder widget. As its name suggests, the StreamBuilder widget is designed to monitor a stream of data for any changes. StreamBuilders have a wide range of uses but are mostly used in cases where there need to be real time updates, as using them can be performance heavy. To enable real-time messaging in The Volunteering Network, a StreamBuilder Widget instance was implemented to observe changes within a specific collection, which in this case represents a conversation. By associating the StreamBuilder with this collection, any updates or modifications are shown instantly.

When a user selects another user to initiate a conversation, the system generates a unique document dedicated to that conversation within the Rooms collection. This document contains information such as the user IDs involved in the conversation, the timestamp of the last sent message, and the contents of that last message. Additionally, a subcollection is created where each message is represented by a separate document. Every time a message is sent, a document is added to that subcollection.

By combining these two concepts and assigning StreamBuilders to monitor specific collections and sub-collections, real time messaging is enabled. This implementation allows for dynamic and immediate updates, without introducing any additional dependances within the application. The Figure 5.7 shows the sequence diagram for the use case of sending a message.

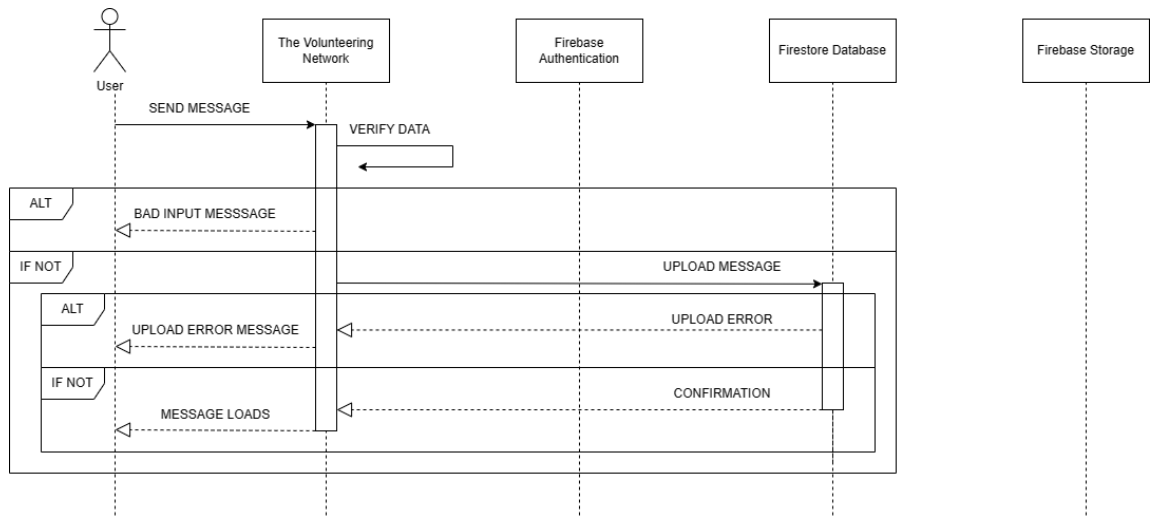


Figure 5.7: Sequence diagram for sending a message

### 5.2.5 Database

Figure 5.8 illustrates the database schema for The Volunteering Network. This figure is added for the purpose of having a visual representation of the relationships within the database.

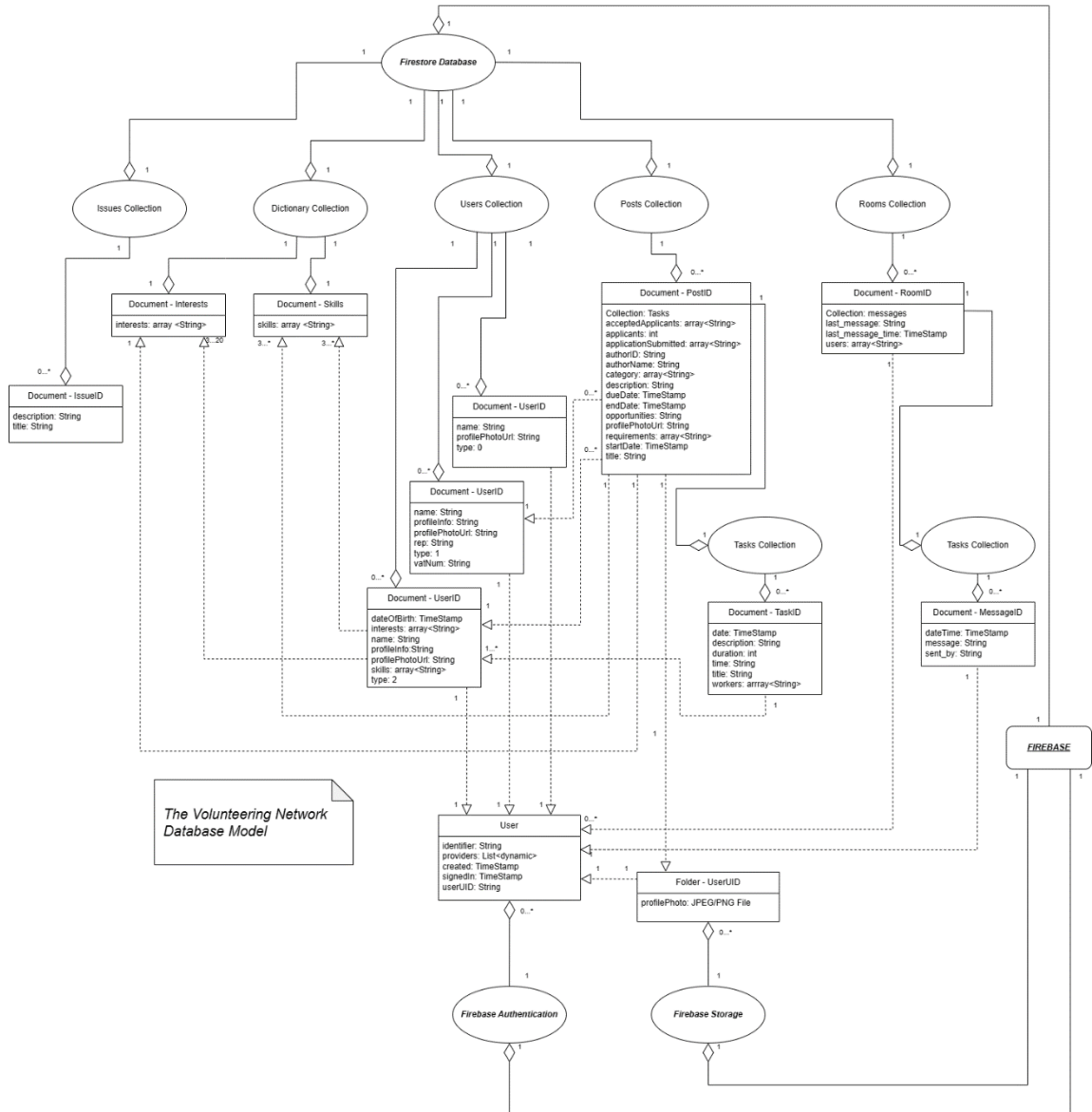
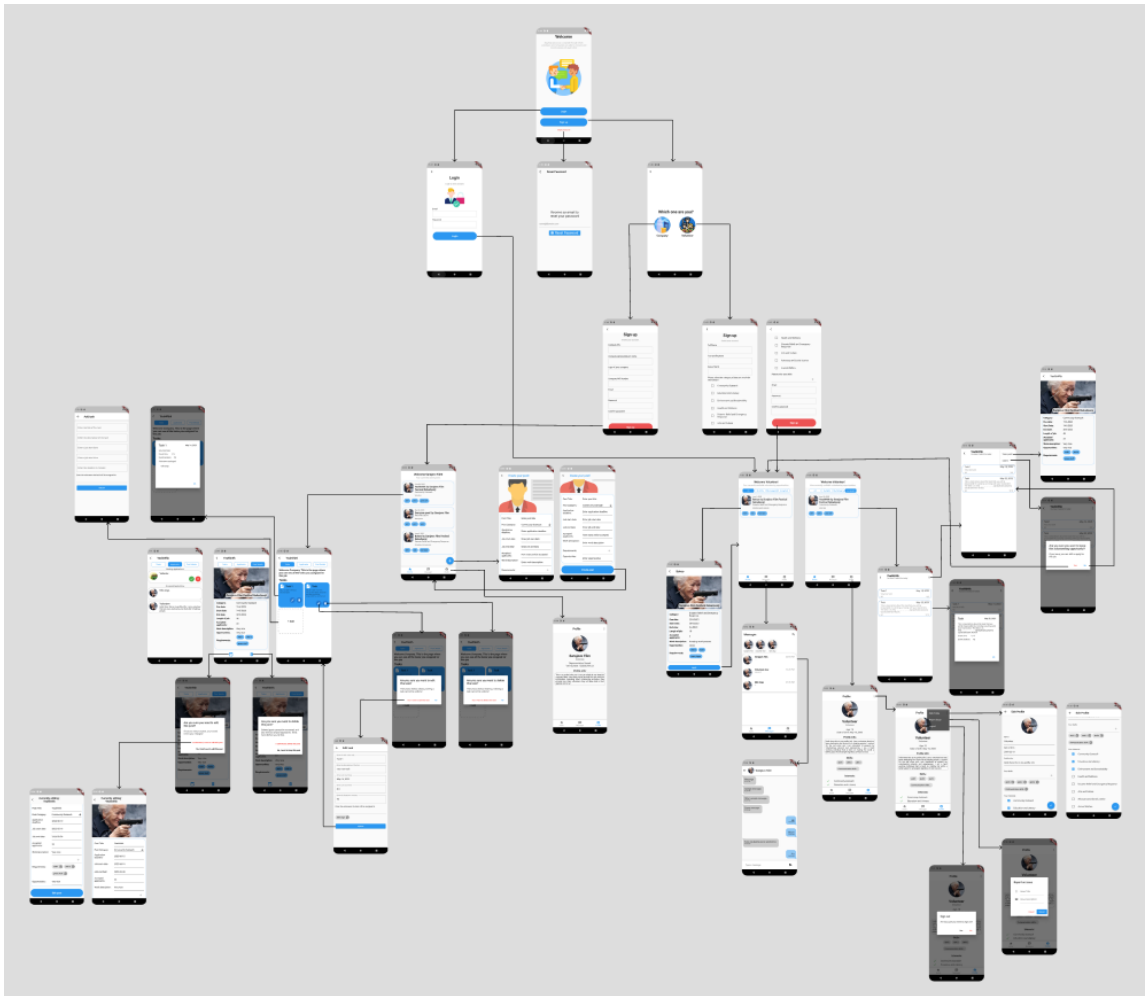


Figure 5.8: Database schema

### 5.3 USER INTERFACE

From the perspective of the average end-user, the user interface represents the most important aspect of an application as it is the part with which they interact with. The UI is one of the parts which can make or break the end-user's wish to use it. Therefore, while designing the application, decisions were made to ensure consistency and clarity

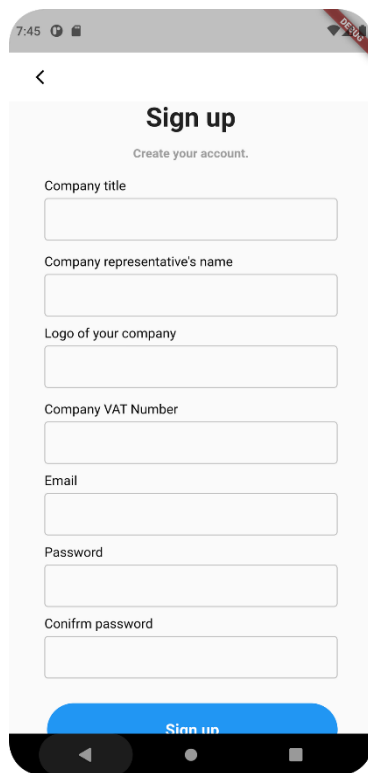
among the various pages and options within the application. The starting point was always to design the application in a minimalist way as much as possible, in order to cater to a larger variety of users. Additionally, the application provides different a slightly different interface for the Volunteer and Company Representative users, based on their respective requirements. To better understand the implementation, four selected pages are presented and explained. Those pages are the sign-up page, dashboard, post creation page and task management page. In Figure 5.9 we can see a view of all of the pages within the application.



*Figure 5.9: Diagram of all pages in application*

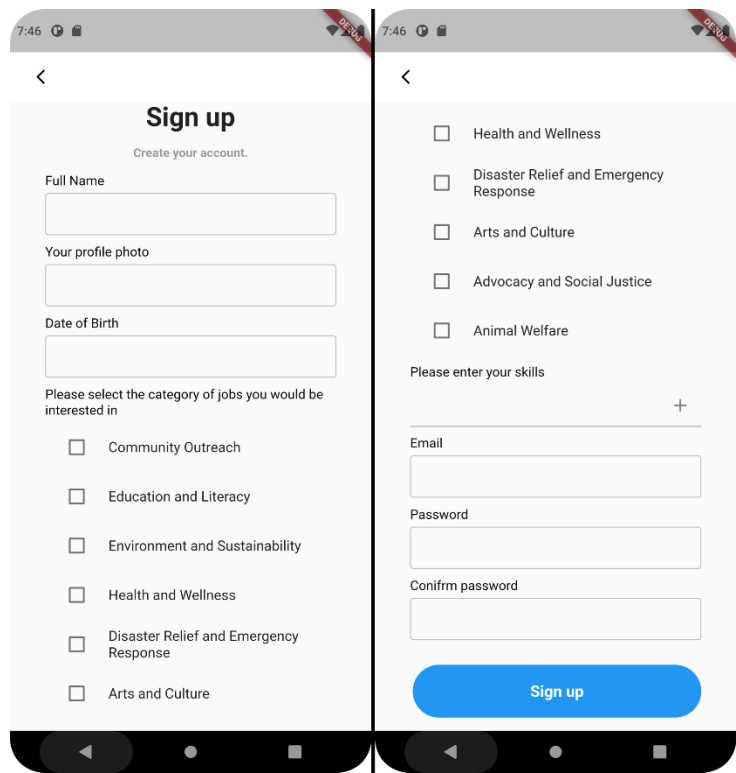
### 5.3.1 Sign-up page

In Figure 5.10 and Figure 5.11 we are able to see the two variations of the sign-up page and how they differ to each other. Each type of user requires different information to be given in order to create the account. To simplify this process, fields such as skills and interests are presented as tags and checklist elements, respectively, in order to reduce the hassle of a completely manual entry of the aforementioned fields. Additionally, every user that is entering their information, has it displayed at all times in case of any potential corrections, as everything entered is considered to be sensitive information.



The screenshot shows a mobile app interface for a volunteer sign-up page. At the top, there is a back arrow and a status bar showing 7:45. The title "Sign up" is centered, with the subtitle "Create your account." below it. The form contains several input fields: "Company title", "Company representative's name", "Logo of your company", "Company VAT Number", "Email", "Password", and "Confirm password". A blue "Sign up" button is at the bottom.

*Figure 5.10: Volunteer user sign-up page*



The screenshot shows a mobile app interface for a company representative sign-up page. At the top, there is a back arrow and a status bar showing 7:46. The title "Sign up" is centered, with the subtitle "Create your account." below it. The form contains several input fields: "Full Name", "Your profile photo", "Date of Birth", "Please select the category of jobs you would be interested in" (with a list of checkboxes: Community Outreach, Education and Literacy, Environment and Sustainability, Health and Wellness, Disaster Relief and Emergency Response, Arts and Culture), "Please enter your skills" (with a plus icon), "Email", "Password", and "Confirm password". A blue "Sign up" button is at the bottom.

*Figure 5.11: Company Representative user sign-up page*

### 5.3.2 Dashboard

The dashboard, shown in Figure 5.12 and Figure 5.13, is used as a location for the posts that are created by companies. One thing to note is how the dashboards differs based on the user type. The Company Representative user may only view the posts which are created by them, without access to any other posts. On the other hand, the Volunteer user is able to view all posts. Furthermore, they are able to filter the posts based on four categories: by skills, by interests, all posts and posts they were accepted to. All posts show only relevant information on the dashboard, as to not overwhelm the user with every detail at the base level.



Figure 5.12: Company Representative user dashboard

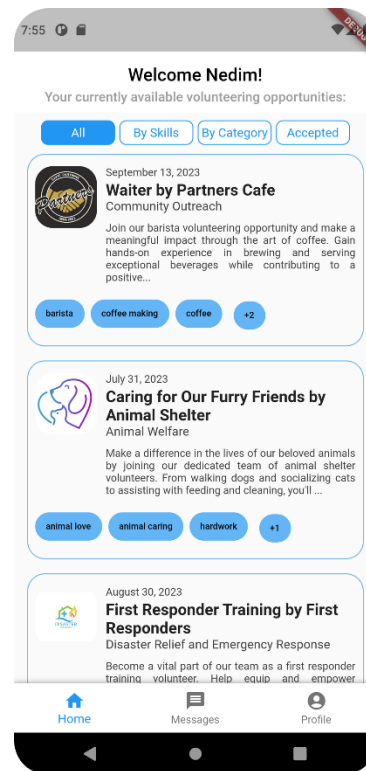


Figure 5.13: Volunteer user dashboard

### 5.3.3 Post creation

Specific only to Company Representative users, the post creation page showcases the details that each post must contain for it to be published. Along with some general information such as the title, description and due date, the Figure 5.14 showcases variety of fields, of which two are crucial for volunteers finding fitting opportunities among the sea of posts, and they are the category and requirements. These two fields translate to the skills and interests the volunteers added to their profile and matches them for quick recommendations to the volunteers (i.e., the aforementioned by skills and by interests filtration on the dashboard). Every field is formatted depending on the type of information that must be entered. While most have a fixed size, some field will expand based on the quantity of information that is to be entered.

The screenshot displays a mobile application interface for creating a post. At the top, there is a status bar showing the time 7:49 and battery level. Below the status bar is a navigation bar with a back arrow and the text "Create your post!". The main content area contains a form with the following fields:

- Post Title: Enter post title
- Post Category: Community Outreach (with a dropdown arrow)
- Application deadline: Enter application deadline
- Job start date: Enter job start date
- Job end date: Enter job end date
- Accepted applicants: How many will be accepted
- Work description: Enter work description
- Requirements: (with a plus icon)
- Opportunities: Enter opportunities

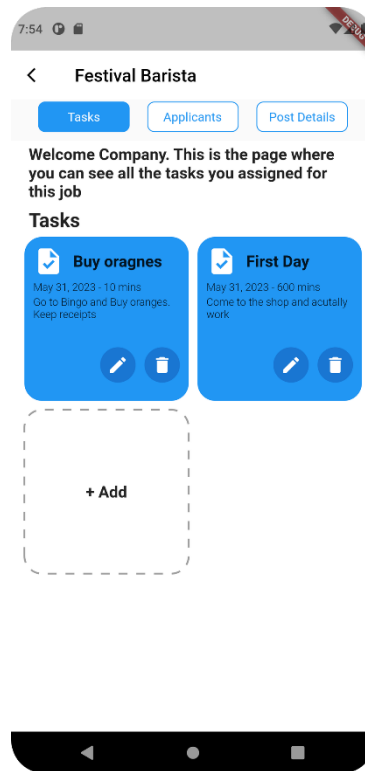
At the bottom of the form is a large blue button labeled "Create post". The interface is clean and modern, with a white background and blue accents.

Figure 5.14: Post creation page

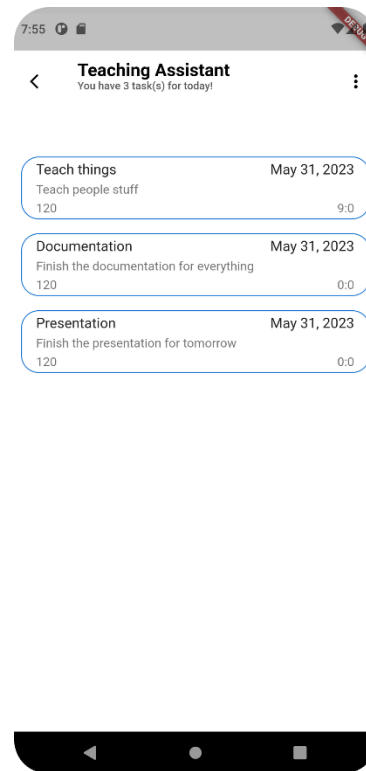


### 5.3.4 Task management

For proper progress tracking, task management was implemented differently for the Volunteer users and Company Representative users, which is showcased in Figure 5.15 and Figure 5.16. The former only have the ability to view the tasks assigned to them, while the latter are the ones who completely manage the tasks for a specific engagement. In addition to basic information such as the title, description and due date, each task is assigned to a Volunteer user upon creation. After it has been created, the figures feature how the tasks shows up for each user. A simplistic design language was used in order to follow the tasks easily and not lose track due to the complexity of the design.



*Figure 5.15: Company Representative user task view page*



*Figure 5.16: Volunteer user task view page*

## CHAPTER 6: CONCLUSION

The Volunteering Network serves as a tool to bridge the gap between all of those uncut gems of volunteers who cannot find suitable opportunities, and the companies that provide said opportunities but have a hard time recruiting suitable people. This thesis presents one unique solution to this problem, along with its architecture, implementation and design. Users of this platform can be classified into three groups: Volunteer, Company Representative and Admin. Volunteers and Company Representatives vary in a few aspects. An example is the former being able to apply to and view posts, view tasks and personalize their profile with skills and interests, while the latter can completely manipulate both posts and tasks. These are just a few of the features available to them. On the other hand, an Admin user is able to completely manage the accounts of the users and review the issue reports submitted. Platforms such as this provide a time-saving solution for both the volunteer and company, as well as improving on the process as a whole with personalized and simple posts and applications. As a result, more ambitious projects can be taken on and less resources can be spent on recruitment for companies. For volunteers, it translates to less hassle to find opportunities and reduced struggle to gain experience. The focus should be to not only simplify, but improve these sort of tasks in order to shift focus to the execution as much as possible. The Volunteering Network was created with these goals in mind.

## REFERENCES

Baeldung. (2021). *Layered Architecture*. Retrieved from Baeldung:

<https://www.baeldung.com/cs/layered-architecture>

Datadog. (n.d.). *Serverless Architecture Overview*. Retrieved from DATADOG:

<https://www.datadoghq.com/knowledge-center/serverless-architecture/#:~:text=Serverless%20architecture%20is%20an%20approach,storage%20systems%20at%20any%20scale>.

Firebase. (2023). Retrieved from Firebase: <https://firebase.google.com/>

Firebase Authentication. (2023). *Firebase Authentication*. Retrieved from Firebase:

<https://firebase.google.com/docs/auth>

Firebase Realtime Database. (2023). *Firebase Realtime Database*. Retrieved from

Firebase: <https://firebase.google.com/docs/database>

Firebase Tutorials. (n.d.). *Firebase Vs AWS: What Are the Differences?* Retrieved from

Firebase Tutorials: <https://firebase-tutorials.com/firebase-vs-aws/>

Firestore. (2023). *Cloud Firestore*. Retrieved from Firebase:

<https://firebase.google.com/docs/firestore>

Fiverr. (2023). Retrieved from fiverr: <https://www.fiverr.com/>

Flutter. (2023). Retrieved from Flutter: <https://flutter.dev/>

Freelancer. (2023). Retrieved from freelancer: <https://www.freelancer.com/>

idealist. (2023). Retrieved from idealist: <https://www.idealists.org/en>

LinkedIn. (2023). Retrieved from LinkedIn: <https://www.linkedin.com/>

TechTarget. (2021). *3-tier application architecture*. Retrieved from TechTarget:

<https://www.techtarget.com/searchsoftwarequality/definition/3-tier-application>

Upwork. (2023). Retrieved from upwork: <https://www.upwork.com/>

VolunteerMatch. (2023). Retrieved from VolunteerMatch:

<https://www.volunteermatch.org/>